Design of a small unmanned aerial vehicle
Group Component

Alp Alptekin, Marco Soto, Eli Bennett, Braden Cardamone, Ben Folgers, Nicolas Vincent

Introduction

The use of Unmanned Aerial Vehicles has been at the heart of aero vehicle development as it was started over a century before the first flight of man. UAVs have a great number of applications in both military and civilian circles. The main uses of UAVs have been:

- Search and Rescue Missions: Military applications of UAVs include surveillance, dropping aid packages, or use of artillery. It is a very effective military technology as it does not put human lives at risk. Furthermore, the use of surveillance allows ground troops to be more aware of the environment, leading to even fewer casualties and a more efficiently run mission.
- Aerial Photography and Videography: UAVs can be used for taking photos and videos at locations that are impossible for a human with a handheld camera. This is mostly applicable to the photography and videography of large buildings, natural structures, or landscapes that can only be captured at a high altitude.
    - Agricultural Purposes: U.S. agricultural land requires effective monitoring of large areas, and the use of drones could assist in this task. It would help to ameliorate farmland issues like pests and fires. Overall imagery for data purposes of the nation's Agriculture is also crucial for government records.
    - Landmark Imagery: Many big cities and famous landmarks around the world contain large structures where imagery is desired and the only product that can fulfill this demand is drones.
- Delivery: A new and useful application of UAVs has been the delivery of groceries, packages, and pharmaceuticals. This provides a lot of value to people, especially due to COVID-19, where no contact transactions are effective. Moreover, for pharmaceutical purposes, it is ideal to get individuals medication without them leaving safe areas such as their homes.

Our project goal is to design an aircraft to deliver medical components in urban areas.

Mastering Airfoil Design

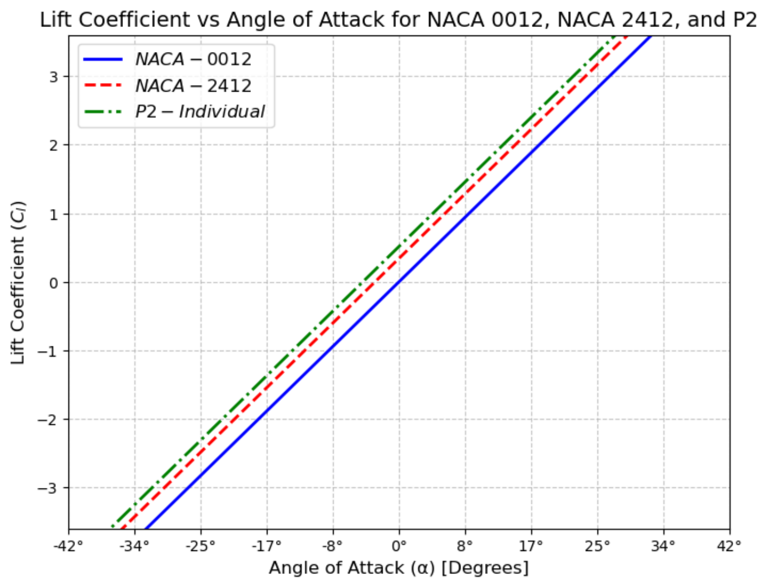

Fig. 1. Graph of C_l vs α for 3 different airfoils

NACA0012

For the NACA 0012, it is a symmetric airfoil, meaning it has no camber. Therefore:

- Maximum Camber: 0 (no curvature).
- Position of Maximum Camber: Undefined, as there is no camber line.
- Angle of Zero Lift: Since there's no camber line, the zero-lift angle of attack for the NACA 0012 is exactly $\alpha = 0$

P2 Individual Component Airfoil

From the P2 individual component airfoil, the camber line function is given as:

$$\frac{z_{c_1}}{c} = \epsilon\left[1 - 16\left(\frac{x}{c} - \frac{1}{4}\right)^2\right], \; for\; 0 \leq \frac{x}{c} \leq \frac{1}{4}$$

$$\frac{z_{c_2}}{c} = \epsilon\left[1 - \frac{16}{9}\left(\frac{x}{c} - \frac{1}{4}\right)^2\right], \; for\; \frac{1}{4} \leq \frac{x}{c} \leq 1$$

For this project our epsilon value, ϵ, has been chosen for us and equals 0.03. Using Python, we can plot the maximum camber and its position along the chord line, both normalized by the chord length shown below:

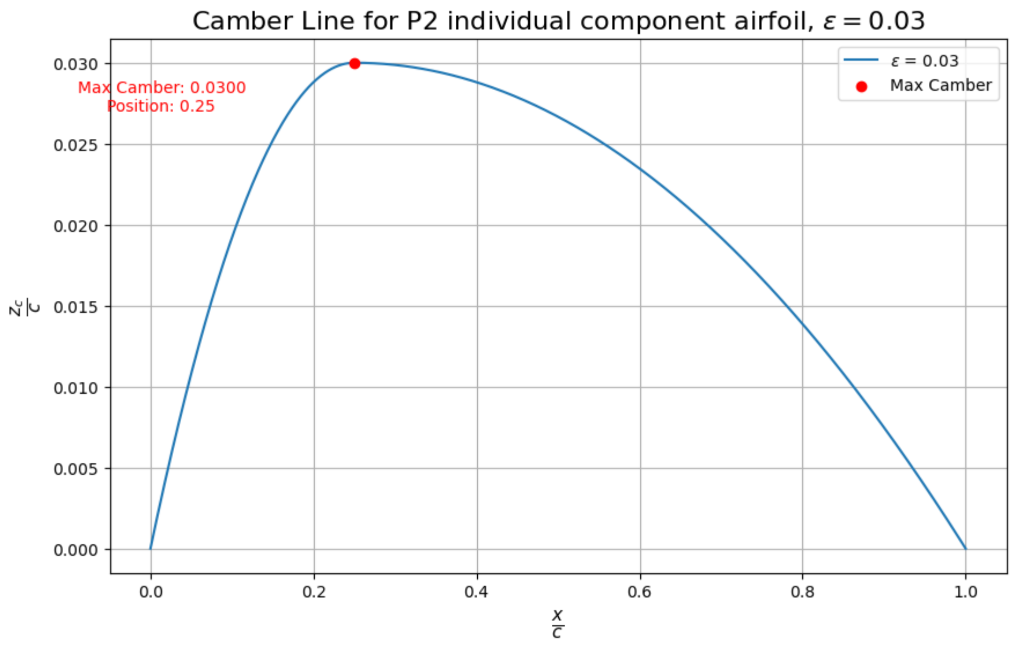

Fig. 2. Plot of maximum camber and position along the chord line for P2 individual component airfoil

We can model our P2 airfoil mean camber line functions as piecewise functions at x = 0.25c. We then take the derivatives of our mean camber line function(s) with respect to x, and we get the following:

$$\frac{dz_{c_1}}{dx} = -\frac{0.96x}{c} + 0.24$$

$$\frac{dz_{c_2}}{dx} = -\frac{0.1066x}{c} + 0.0267$$

We then make the following substitution to transform our functions into polar form:

$$x = \frac{c}{2}(1 - \cos(\theta))$$

So then:

$$\frac{dz_{c_1}}{dx} = -0.48(1 - \cos(\theta)) + 0.24$$

$$\frac{dz_{c_2}}{dx} = -0.0533(1 - \cos(\theta)) + 0.0267$$

$$\frac{c}{2}(1 - \cos(\theta)) = 0.25c$$

$$\theta = 1.0472\ rad$$

From Thin Airfoil Theory (TAT):

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

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

$$C_l = 2\pi\left(A_0 + \frac{A_1}{2}\right)$$

Using our previous piecewise functions, we can break up these integrals and solve for our zero-lift angle for the P2 airfoil as follows:

$$A_0 = \alpha - \frac{1}{\pi}\int_0^{1.0472} (-0.48(1 - \cos(\theta)) + 0.24) d\theta - \frac{1}{\pi}\int_{1.0472}^{\pi} (-0.0533(1 - \cos(\theta)) + 0.0267)\, d\theta$$

$$A_0 = \alpha - 0.019893$$

$$A_1 = \frac{2}{\pi}\int_0^{1.0472}(-0.48(1-\cos(\theta))+0.24)\cos(\theta)d\theta + \frac{2}{\pi}\int_{1.0472}^{\pi}(-0.0533(1-\cos(\theta))+0.0267)\cos(\theta)\,d\theta$$

$$A_1 = 0.136693$$

$$C_l = 2\pi\alpha + 0.304441$$

We know that $C_l = 0$ when $L' = 0$, therefore we can set the left hands side of our equation to zero and solve for α_0, our zero-lift angle of attack:

$$0 = 2\pi\alpha + 0.304441$$

$$\alpha_0 = -0.048453\,rad = -2.7762^0$$

NACA2412

For the NACA 2412 airfoil, the camber line function is given as:

$$\frac{z_{c_1}}{c} = 0.125\left[0.8\left(\frac{x}{c}\right)-\left(\frac{x}{c}\right)^2\right],\ for\ 0 \le \frac{x}{c} \le 0.4$$

$$\frac{z_{c_2}}{c} = 0.05555\left[0.2+0.8\left(\frac{x}{c}\right)-\left(\frac{x}{c}\right)^2\right],\ for\ 0.4 \le \frac{x}{c} \le 1$$

Once again, we can plot the maximum camber and its position along the chord line using Python, both normalized by the chord length shown below:

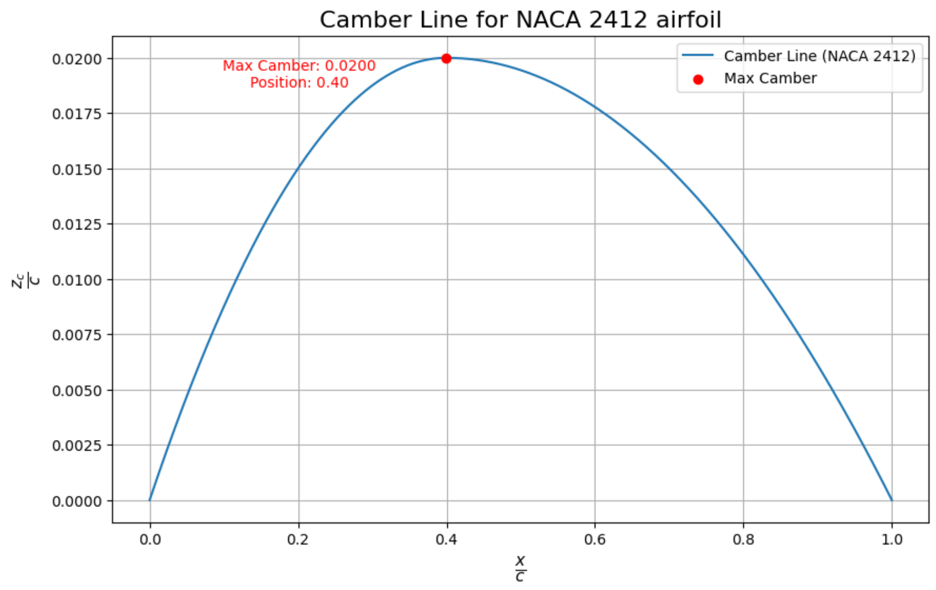

Fig. 3. Plot of maximum camber and position along the chord line for NACA 2412 airfoil

We can also model the NACA 2412 airfoil mean camber line functions as piecewise functions at x = 0.4c. However, we are given the following values of A_n so computing the zero-lift angle will be much easier:

$$A_0 = \alpha - 0.0044929$$

$$A_1 = 0.081495$$

$$A_2 = 0.013861$$

From Thin Airfoil Theory (TAT):

$$C_l = 2\pi\left(A_0 + \frac{A_1}{2}\right)$$

$$C_l = 2\pi\alpha + 0.304441$$

We know that $C_l = 0$ when $L' = 0$, therefore we can set the left-hand side of our equation to zero and solve for α_0, our zero-lift angle of attack for the NACA 2412 airfoil:

$$0 = \ 2\pi\alpha + 0.22779$$

$$\alpha_0 = \ -0.036255\, rad = \ -2.077^0$$

Table 1. Table of the maximum camber and its position along the chord line, both normalized by the chord length, the angle of zero lift and the lift slope for all three airfoils.

Airfoil	Max. camber (normalized)	Position of max. camber (normalized)	α_0	Lift slope
NACA 0012	0	N/A (0)	0°	2π
P2 Individual	0.03	0.25	-2.776°	2π
NACA 2412	0.02	0.4	-2.077°	2π

All three airfoils follow the linear lift curve slope of 2π, as predicted by Thin Airfoil Theory (TAT). This reflects their behavior in the small angle of attack pattern, which states that lift is proportional to the angle of attack. The NACA 0012 airfoil has $\alpha_0 = 0°,$ reflecting its symmetric design. The P2 individual airfoil, with a maximum camber of 0.03 at 25% chord, has the most negative zero-lift angle $\alpha_0 = -2.776°$, indicating significant camber. The NACA 2412 airfoil, with a lower camber of 0.02 at 40% chord, has a less negative zero-lift angle of $\alpha_0 = -2.077°$. The differences in camber affect stall behavior and aerodynamic efficiency. The NACA 0012 must always have a positive angle of attack to generate lift. The P2 airfoil may generate a higher lift at lower angles of attack due to its higher camber, whereas the NACA 2412 offers a compromise between the other 2 airfoils.

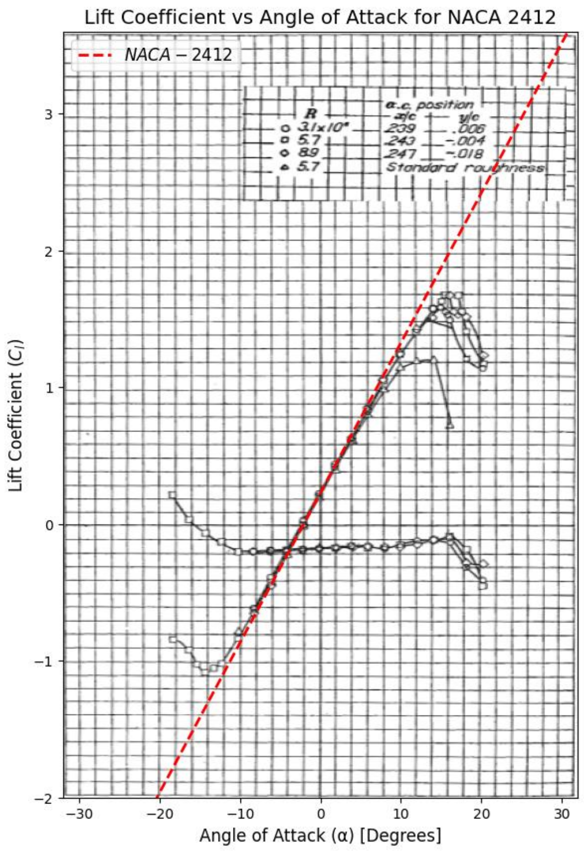

Fig. 4. Combined plot of lift coefficient vs angle of attack for experimental and theoretical NACA 2412 airfoil

Both curves follow a nearly identical linear slope in the low angle of attack range $(-8^\circ \leq \alpha \leq 8^\circ)$, which is consistent with Thin Airfoil theory expectations. The agreement between the curves in this range indicates that the theoretical model effectively captures the relationship between the angle of attack and lift coefficient for small angles. Also, the zero-lift angle of attack is very similar in both curves $(\alpha \approx -2^\circ)$, matching theoretical expectations for a cambered airfoil like the NACA 2412. However, the experimental curve shows a noticeable drop/increase in C_l after reaching the stall angle (roughly $\pm 16^\circ$) representing flow separation. In contrast, the theoretical curve continues to increase linearly, as it fails to account for non-linear, real-world effects like boundary layer separation and turbulence.

We recall the Reynold's Number equation:

$$R_e = \frac{\rho v L}{\mu}$$

Based on the parameters from the P2 individual component, and the dynamic viscosity at sea level, we substitute them into the previous equation and solve for the theoretical Reynold's Number:

$$R_e = \frac{\left(1.225\,\frac{kg}{m^3}\right)\left(10\,\frac{m}{s}\right)(0.437m)}{1.789 \times 10^{-5}\left(\frac{kg}{m * s}\right)}$$

$$R_e = 299231.414$$

At a R_e of ~299,231 the flow is predominantly inviscid and may transition into turbulent downstream. Due to its large magnitude, viscous effects are less dominant, but they still influence flow behavior, especially near stall, causing deviations from the theoretical curve. Higher R_e would result in more turbulent flow over the surface, further delaying separation and reducing these differences. This calculated R_e explains why the experimental data aligns well in the pre-stall region but diverges near and beyond stall, as it is in the moderate range for the parameters given.

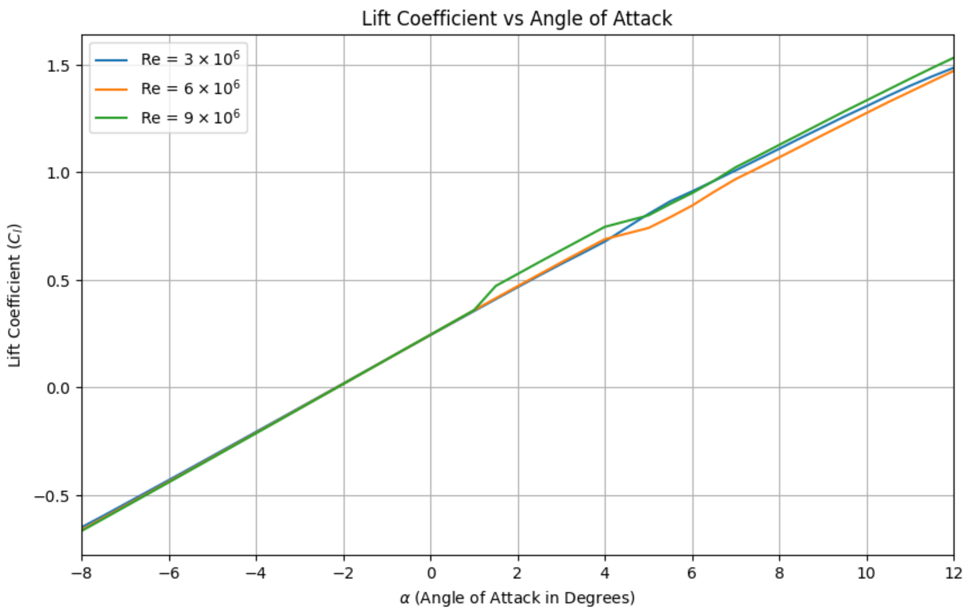

Fig. 5. Plot of lift coefficient vs angle of attack for NACA 2412 airfoil at varying Reynolds Numbers

The general trend for all Reynolds Numbers appears to be linear, showing an increase in lift coefficient as the angle of attack increases. Additionally, all three curves intersect at similar established points, such as the zero-lift angle (roughly -2°) which indicates the airfoil performance remains consistent in certain flow conditions. However, the slopes slightly vary and begin to deviate from one another as the plot approaches positive alpha values, an indicator of lift-generation differences due to varying Reynolds Numbers. As seen in the plot, lower Reynolds Numbers exhibit more consistent lift performance while higher Reynold's Numbers improve lift performance but start to deviate from theoretical assumptions and display fluctuation at higher angles of attack.

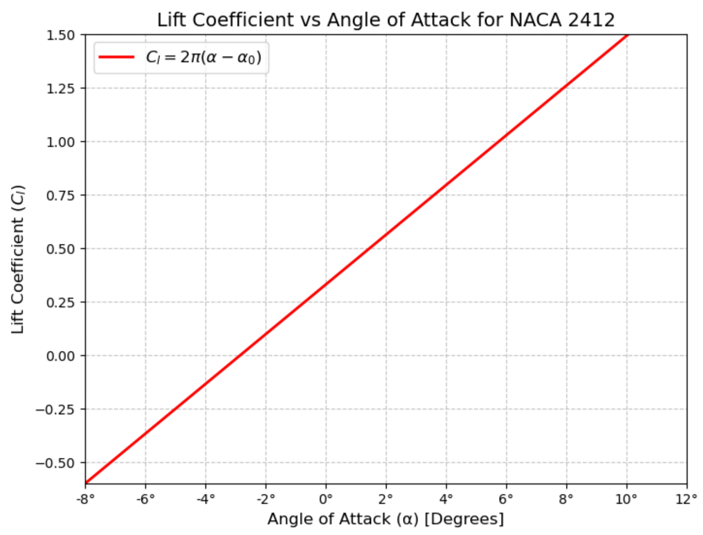

Fig. 6. Plot of lift coefficient vs angle of attack for NACA 2412 airfoil

When comparing the plot obtained with vortex panel method and Thin Airfoil Theory, we see that they both follow a linear relationship between C_l and α at low angles of attack, further reflecting the applicability of TAT (assuming ideal conditions). Hence, TAT provides a reasonable prediction when we neglect other factors such as viscous effects, Reynolds Numbers, and stall. Additionally, it's worth noting that the theoretical plot (Fig. 6) most closely resembles the plot for the NACA 2412 at $R_e = 3 \times 10^6$. This could be due to the lower Reynolds Number capturing flow behavior that aligns more closely with the assumptions of thin airfoil theory, such as negligible flow separation at low angles of attack.

Designing the wing

In designing the wing, this includes doing and testing mathematics of a variety of wings and running simulations and graphs to determine which wing is most efficient for our aircraft's desired purpose.

The formula that will be used to determine the geometric angle of attack with direct relation to zero-lift angle of attack, lift coefficient, and lift slop is:

$$C_L = a(\alpha - \alpha_{Lo}),\ \alpha = \frac{C_L}{a} + \alpha_{Lo}$$

In order to calculate the geometric angle of attack in the different wing types, we must determine lift coefficient, lift slope, and zero-lift angle of attack. These will be different in all wings as they have different shape, effects, etc.

We start off with determining the base for the comparison of wings, which will be a two-dimensional airfoil of infinite span with the same chord used in the individual components' preliminary rectangular wing. The geometric angle of attack and induced drag coefficient will be calculated for this wing. The angle of attack will showcase the required angle to generate sufficient lift at this base wing. The induced drag coefficient will display the overall lift efficiency. Induced drag is caused through the generation of lift and is unavoidable, however the mitigation of its impact on aircraft flight can be achieved. Below are the calculations to determine the variables required to ultimately compute the geometric angle of attack and induced drag coefficient.

$$C_L = \frac{L'}{0.5 * \rho V_\infty^2 s}$$

$$L' = F_w = mg = 6.2 \cdot 9.81 = 60.822$$

$$\rho = 1.225 \frac{kg}{m^3}, V_\infty = 10 \frac{m}{s}, s = lw = 2 \cdot 0.43 = 0.86$$

$$AR = \frac{w^2}{s} = \frac{2^2}{0.86} = 4.65$$

$$C_L = \frac{60.822}{\frac{1}{2}(1.225)(10^2)(0.86)} = 1.155$$

$$a = a_o = 2\pi$$

To calculate α_{Lo}:

To calculate the zero-lift angle of attack, some assumptions that can be made are that lift coefficient is 0 as there is no produced lift. Through the calculated Fourier series coefficients, the zero-lift angle of attack can be computed as such:

$C_L = 2\pi A_o + \pi A_1$, and when $\alpha = \alpha_{Lo}, C_L = 0$ $0 = 2\pi(\alpha - 0.0044929) + \pi(0.081495) = \alpha = \alpha_{Lo} = -0.0407475 + 0.0044929 = -0.0362546 = -2.077^\circ$

Now that lift coefficient, lift slope, and zero-lift angle of attack at a two-dimensional airfoil are known, the geometric angle of attack can be solved:

Two-Dimensional Airfoil

$$\alpha = \frac{1.155}{2\pi} - 0.0362546 = 0.14756936\, rad = 8.455^\circ$$

Induced Drag Coefficient

A circumstance of the two-dimensional airfoil with infinite span is that due to its infinite span, the Aspect Ratio of the wing is infinity, which would lead to an induced drag coefficient of 0, as well as lift coefficient being 0 as there is no lift produced:

$$C_{Di} = \frac{C_L^2}{\pi(\infty)} = 0$$

Elliptical Wing

After quantifying the required values for the two-dimensional airfoil with infinite span, which was the base for comparison, it can now be presumed that further calculation will include the finite effects of 3D airfoils. Due to this, less assumptions can be made, which leads to increased calculation for certain values.

$$\alpha = \frac{C_L}{a} + \alpha_{Lo}$$

Lift Slope is now not at a constant value of 2π as now the 3D finite effects are present and in the situation of Elliptical Lift Distribution, the lift slope formula is as follows:

$$a = \frac{a_o}{1+\frac{a_o}{\pi AR}},\ AR = \frac{w^2}{s} = \frac{2^2}{0.86} = 4.65,\ a_o = 2\pi$$

$$a = \frac{2\pi}{1+\frac{2\pi}{\pi(4.65)}} = 4.393$$

Using the new calculated lift slope, we can now determine the geometric angle of attack for an elliptical wing:

$$\alpha = \frac{C_L}{a} + \alpha_{Lo} = \frac{1.155}{4.393} + -0.0362546 = 0.227 = 13.006^o$$

Now, unlike the two-dimensional airfoil, the induced drag coefficient can now be calculated as there is an existent aspect ratio that isn't infinite. This value was calculated earlier, in order to compute the lift slope and is used to solve the induced drag coefficient as follows:

$$C_{Di} = \frac{{C_L}^2}{\pi AR} = \frac{1.155}{\pi(4.65)} = 0.0913$$

Root Chord

For the comparison of all the wings that will be considered it is important to determine basic dimensional values and one of great importance will be the root chord. Through graph analysis, we will observe if there is correlation between the root chord length and efficiency of flight. To calculate the root chord of the elliptical wing, we will be able to use the formula:

$$c_{root} = \frac{4S}{\pi b} = \frac{4(0.86)}{\pi(2)} = 0.6475$$

Designing a Tapered Airfoil

In order to maximize efficiency and simplify calculations, our team selected a taper ratio $\lambda = \frac{1}{3}$. This falls within the 0.3-0.4 range which minimizes the value of the slope parameter, τ. Calculating the chord length at the root and tip, c_{root} and c_{tip}, we used the equation:

$c_{root} = \frac{2S}{b} - c_{tip}$, given inputs: $S = 0.86,\ b = 2m, \rightarrow \lambda = 0.33$

With these inputs, we can calculate:

$$c_{root} = \frac{2S}{b(1+\lambda)} = \frac{2(0.86)}{2(1+0.33)} = 0.647m$$

Tapered Airfoil Analysis:

Next, we set out to calculate the slope, A, the coefficient of induced drag, C_{D_i}, and the required angle of attack at level flight, α, for our tapered wing design. To begin we utilized the equation:

$A = \frac{a_0}{1+\frac{a_0}{\pi AR}(1+\tau)}$, with inputs: $a_o = 2\pi$, $AR = \frac{s^2}{A} = \frac{4}{0.86} = 4.65$, and $\tau = 0.025$ (graphically determined),

With these values we determined $A = 4.361$ for the tapered wing.

Using the equation: $C_{D_i} = \frac{{C_L}^2}{\pi AR}(1+\delta)$, with inputs: $C_L = 1.155$ (previously calculated), $AR = 4.65$, and $\delta = 0.01$(graphically determined) we calculated: $C_{D_i} = 0.0922$ for the tapered wing.

Finally, we can calculate α using the equation: $\alpha = \frac{C_L}{A} + \alpha_{L=0}$ given inputs: $C_L = 1.155$, $A = 4.361$, and $\alpha_{L=0} = -2.077^{\circ}$(previously calculated). Computing this we find $\alpha = 13.1°$ for the tapered airfoil.

Rectangular Airfoil Analysis:

The process for calculating the A and C_{D_i} for a rectangular airfoil is the same as the tapered airfoil except the τ and δ are determined graphically using a taper ratio of 1. Utilizing the equation:

$A = \frac{a_0}{1+\frac{a_0}{\pi AR}(1+\tau)}$, with inputs: $a_o = 2\pi$, $AR = 4.65$ (previously calculated), and $\tau = 0.15$, we calculated $A = 4.20$ for the rectangular wing.

Using the equation: $C_{D_i} = \frac{{C_L}^2}{\pi AR}(1+\delta)$, with inputs: $C_L = 1.155$, $AR = 4.65$, and $\delta = 0.025$, we calculate the $C_{D_i} = 0.094$ for the rectangular wing design.

Finally, we can calculate α using the equation: $\alpha = \frac{C_L}{A} + \alpha_{L=0}$ given inputs: $C_L = 1.155$, $A = 4.20$, and $\alpha_{L=0} = -2.077^{\circ}$(previously calculated). Computing this we find $\alpha = 13.68°$ for the rectangular airfoil.

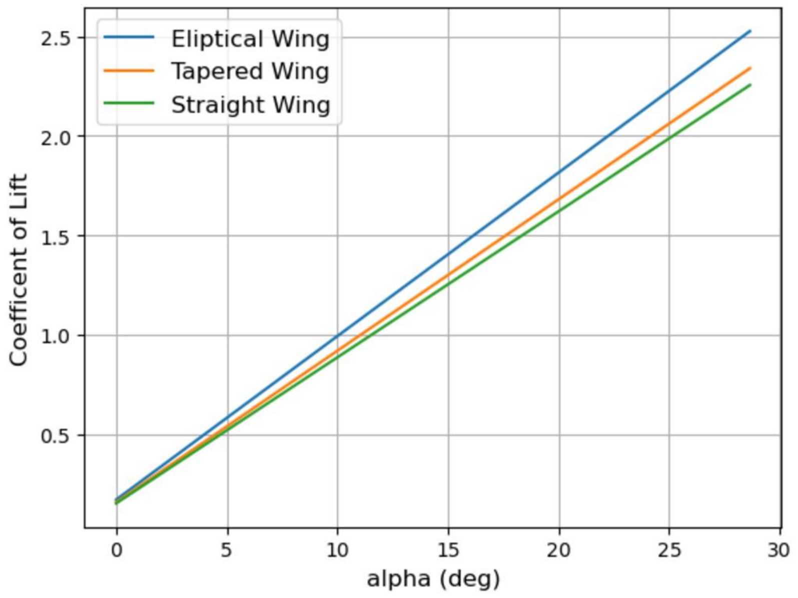

Fig. 7. Combined plot of lift coefficient vs angle of attack for elliptical, tapered, and straight wing

To decide on a wing shape, our team ran a rudimentary trade study deciding on different weights given categories of L/D, cost, and assembly time. L/D was given a weight of 500 to ensure that it is an important factor and to differentiate the somewhat similalar L/D values. Cost and assembly were both given a weight of -20 to penalize them and have a significant effect on the outcome of the trade study. The cost and assembly time were decided qualitatively. The trade study determined for our team that the ideal wing for this application would be a tapered wing due to its higher L/D and efficiency in manufacturing and assembly.

Table 2. Cost and assembly times of different wing shapes

	L/D		Cost		Assembly Time			Total Value
Eliptical	12.6506	1.012995	2	1.428571	3	1.730769		443.3107
Tapered	12.52711	1.003107	1.2	0.857143	1.2	0.692308		470.5644
Straight	12.28723	0.983898	1	0.714286	1	0.576923		466.125
	Weight:		Weight:		Weight:			
	500		-20		-20			

bel

Plotted below are the simulated and theoretical coefficients of lift versus the angle of attack and the coefficients of drag versus the coefficients of lift for wings of different shapes with a NACA 2412 airfoil, using the XFLR5 software.

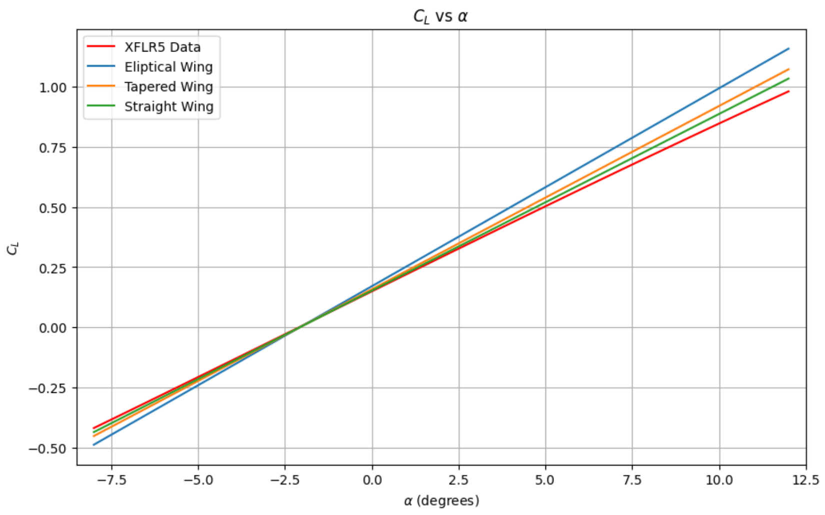

Fig. 8. Combined plot of lift coefficient vs angle of attack for experimental and theoretical NACA 2412 airfoil

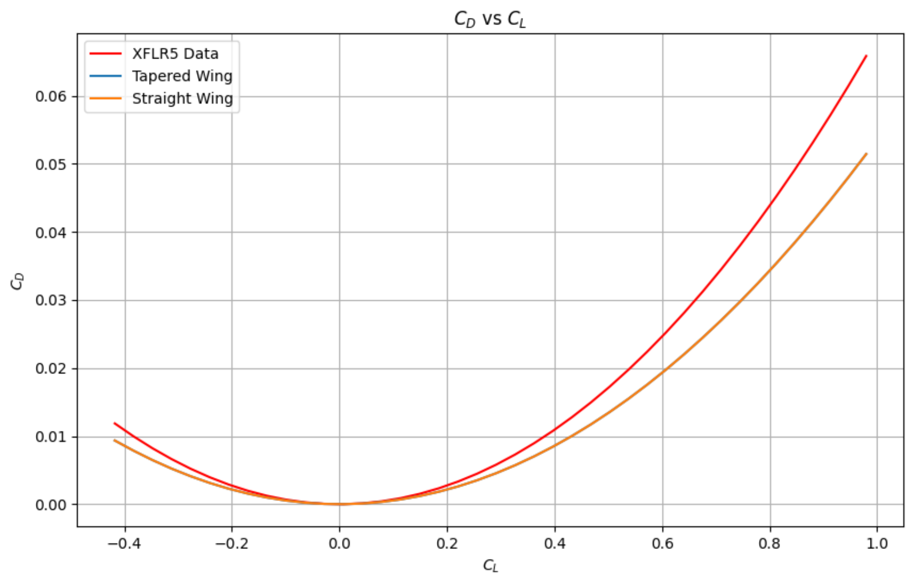

Fig. 9. Combined plot of drag coefficient vs drag coefficient for experimental and theoretical NACA 2412 airfoil

In the $C_L\ vs\ \alpha$ graph, we see very similar results. Most notably, the angle of attack of zero lift is the same across all plots. This means that the zero-lift angle of attack is primarily determined by the airfoil's camber and is unaffected by other factors such as wing geometry. Beyond that, the coefficient of lift does not vary that much between the wing shapes, showing that the overall lift characteristics are relatively insensitive to changes in the wing shapes.

In the $C_L\ vs\ C_D$ graph, we also see a lot of similarities. Due to the very similar induced drag parameter between the tapered and straight wing, the coefficient of drag as a function of the coefficient of lift is nearly identical. Both plots are very similar to the function obtained from the XFLR5 data. This tells us that the aerodynamic efficiency, represented by the lift-to-drag ratio, is consistent across the tested wing shapes and closely aligns with the predictions from the XFLR5 simulation. This suggests that the induced drag is predominantly influenced by the aspect ratio and spanwise lift distribution, which remain similar between the designs.

Advanced Design

Flaps, slats, and ailerons are all control surfaces that are present on the primary wing of an aircraft. While ailerons are primary control surfaces, flaps and slats are secondary control surfaces. All three control surfaces work in similar ways, as they effectively alter the camber of the wing, increasing or decreasing the lift of that section of the wing based on the desired control outcome. Ailerons are primary control surfaces that control the roll of the aircraft. There is one aileron per wing, with both ailerons deflecting in different directions. To induce a moment to roll the aircraft, one aileron deflects upwards, and the other aileron deflects downwards. The aileron that deflects downwards will create more lift as the effective camber of the wing will be higher. This will cause this wing to create more lift. The other wing has an aileron that deflects upwards, which decreases the camber, and thus the overall lift. The wing containing the upwards-deflecting aileron will produce less lift, which will start to roll the aircraft in the direction of this wing. Leading-edge slats and trailing-edge flaps are used to increase the lift of the wing, typically used when landing or taking off to reduce the required speed to create a certain amount of lift. By reducing the amount of speed required to get a certain lift, the takeoff can be shortened, or the approach speed can be decreased, leading to a shorter landing. The flaps have various actuation methods, but on smaller aircraft, the flaps simply take a section towards the trailing edge of the wing and deflect it downwards to increase the camber and thus, increase the lift. Leading-edge slats are typically not common on smaller aircraft, as they are mechanically complex, and not usually necessary for small aircraft. Nonetheless, slats typically deflect downwards on the leading-edge of the wing and extend the overall length of the airfoil by extending forward as well. For the sake of this aircraft, we can simplify our model to not contain slats, to avoid the mechanical complexity and weight gain from the addition of slats. We can instead model the aircraft to have trailing edge flaps, in the form of plain flaps. These flaps can deflect to 0, 10, 20, and 40 degrees. We modeled a wing with flaps taking up ¼ of the total span of the wing, so the flaps are roughly 0.25m in width, with the total length being 20% of the chord length.

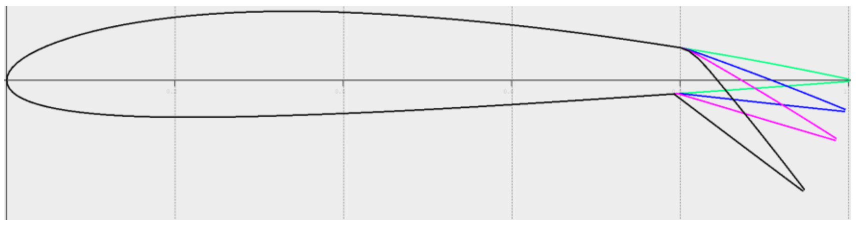

Fig. 10. Side view of different flap profiles

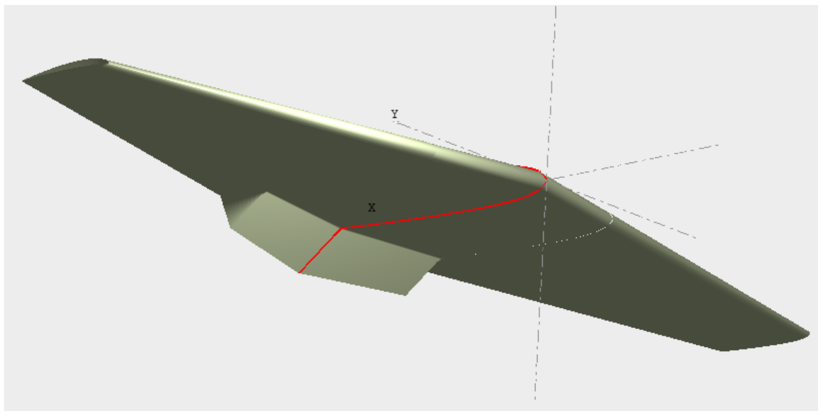

Fig. 11. 3D Wing used to calculate C_L with 40º flaps

Using the XFLR5 software and plotting this as a tapered wing with the flaps at 25% of the span, we obtain the following curves for C_L vs α and C_L vs. C_D.

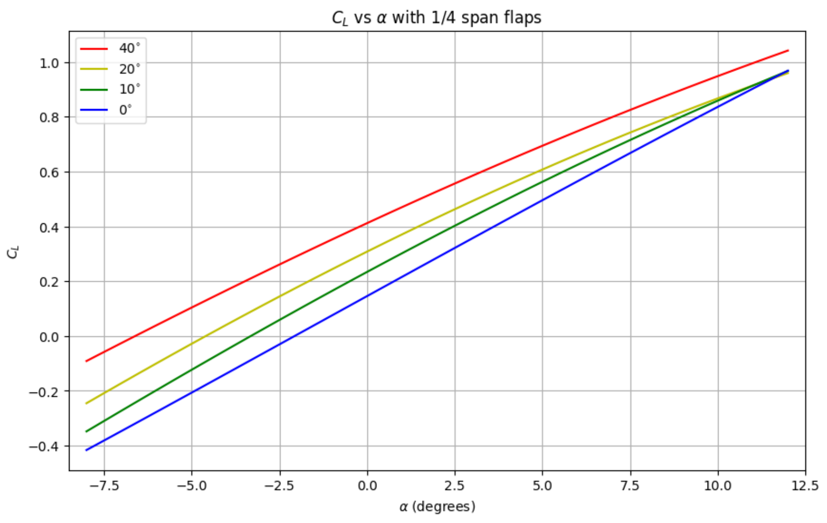

Fig. 12. Combined experimental plot of C_L vs α with flaps at various angles

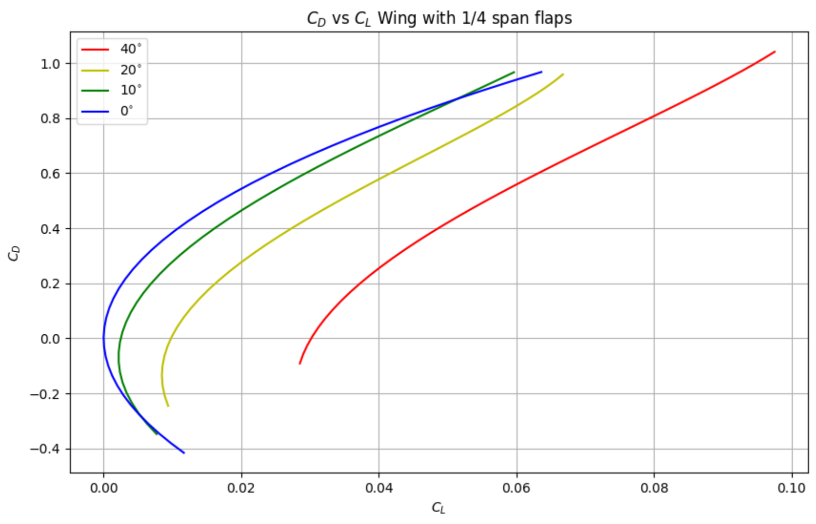

Fig. 13. Combined experimental plot of C_D vs C_L with flaps at different angles

The addition of flaps allows us to obtain a much higher C_L for a given angle of attack. Additionally, the C_D for a given C_L decreases as we increase the flap angle. This indicates that the wing will produce more lift at any given angle of attack, allowing the aircraft to fly more efficiently at lower speeds, especially near takeoff and landing. The same applies when comparing the graphs of C_L vs C_D, as we increase the flap angle, the

To validate the theory of the ailerons, two airfoils were made: one represented the upward deflection of the control surface, and the other represented the downward deflection of the control surface. These were then entered into XFLR5 to analyze how the C_l related to α, which in turn would represent the respective wing lifts for a given angle of attack with the ailerons deflected.

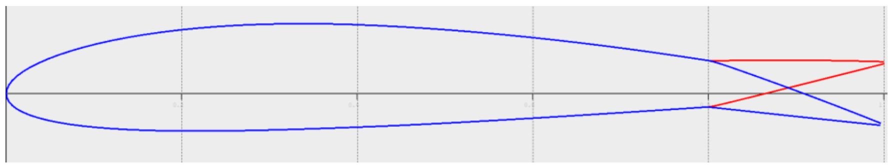

Fig. 14. Side view of aileron airfoils

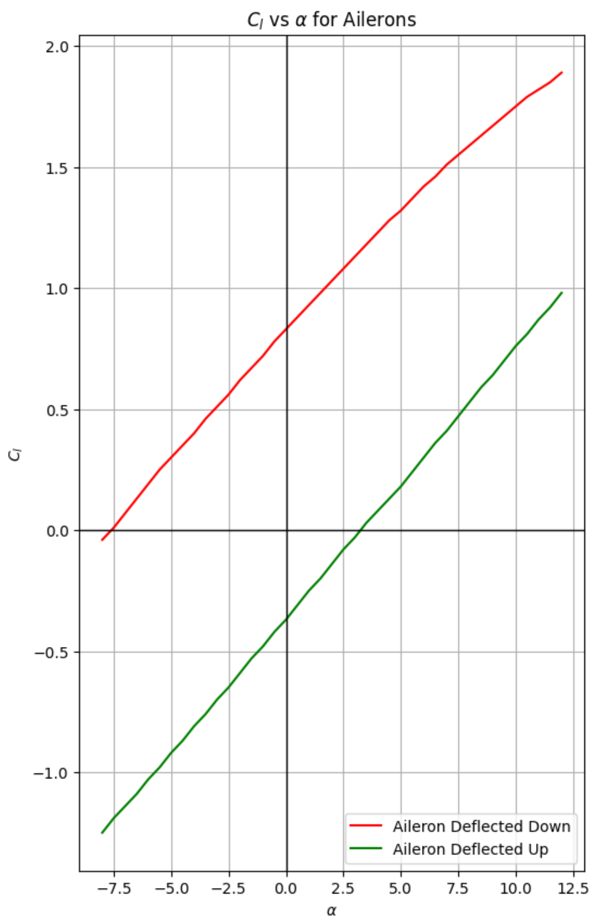

Fig. 15. C_l vs α for aileron deflection

By analyzing the curves for C_l vs α we see that when the aileron is deflected downwards, the wing produces much more lift for a given angle of attack. Conversely, the aileron that is deflected upwards will create less lift, so this wing will drop. This proves the theory as to how ailerons work, as when one aileron is deflected downwards, and when one is deflected upwards it will induce a roll that will allow us to control the aircraft as desired.

As mentioned before, the addition of ailerons is crucial for controlling the roll of the aircraft and should be implemented into the final design of the aircraft. Trailing-edge flaps should be added to the aircraft as well, as it allows the aircraft to take off and land at slower speeds, lowering the effectively required length of the runway. One thing that should not be implemented is leading-edge slats, as the mechanical complexity and weight addition of slats is not beneficial enough to add these devices to the wings. Even on smaller general aviation aircraft, slats are typically uncommon, citing the same mechanical complexity and weight issues as primary reasons they are not implemented.

Writing A Python-Based Circulation Program

To create a Python-based program that can be used to calculate the C_L for a given α, the first goal was to create a program that can be used to calculate a variable number of points along a given 4-digit NACA airfoil. The first program takes two parameters: 4 digits for the desired NACA airfoil and the number of points that are desired to create the spline for the airfoil. Using the following thickness distribution, and spanning our space along the number of points, we can generate the basics for our spline:

$$y = \frac{t}{c}\left(A\sqrt{x} + Bx + Cx^2 + Dx^3 + Ex^4\right)$$

Fig. 16. Thickness distribution for a 4-digit NACA airfoil

This thickness distribution has the following coefficients:

A = 1.4845
B = -0.630
C = -1.758
D = 1.4215
E = -0.5075

Using this thickness distribution for the NACA airfoil, while also parsing the maximum thickness, position of maximum camber, a function was defined that plotted the surfaces for any given NACA airfoil. NACA 2412 was the airfoil of interest; however, the NACA 0012 airfoil was also plotted to ensure that the function was working properly.

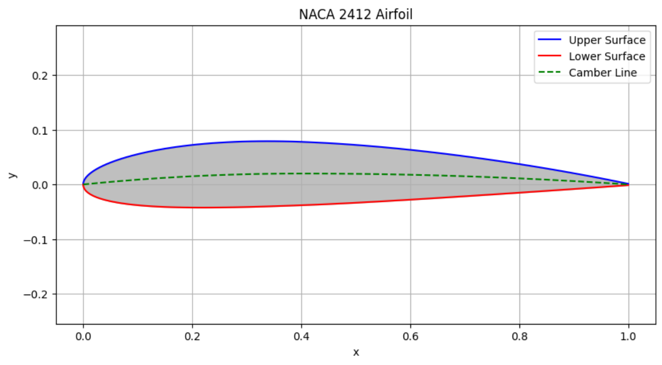

Fig. 17. Python Generated Airfoil for NACA 2412

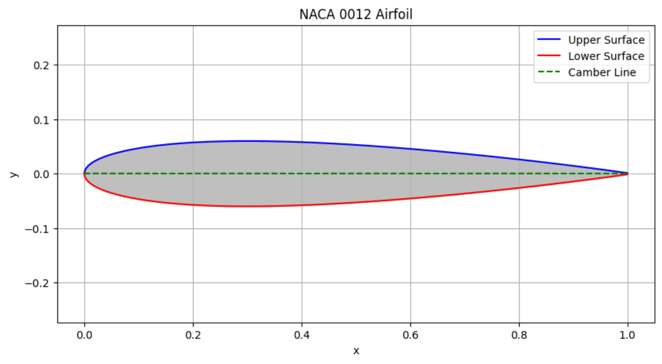

Fig. 18. Python Generated Airfoil for NACA 0012

Once the airfoils were properly plotted, the arrays of the positions of the surfaces of the airfoils were then passed to the function to calculate the C_L vs α. To calculate the proper C_L, a panel was created between every defined point along the airfoil surface, and along this panel, a vortex was defined along 25% of the panel, with a control point being defined 75% along the panel. Once the vortices and control points were determined, the influence matrix (A) was defined. The influence matrix is filled with the values of the individual influences each vortex has on the vortex of interest. Once the influence matrix was defined, we determined the vector (B). The B vector was the vector that represents the angle the panel makes with the angle with the tangential flow. These two are necessary to fulfill the following equation to solve the vortex panel method.

$$\Gamma = A^{-1}B$$

Once the circulation due to the vortex panel method is defined, we can calculate the coefficient of lift by utilizing the following equation.

$$C_L = \frac{2\Sigma\Gamma_i}{V_\infty c}$$

This equation states that if we sum all our circulation and multiply by two, we should get the respective coefficient of lift. Using a normalized velocity V = 1, and a normalized chord length c = 1, the denominator of the equation can be ignored. Summing the values for circulation and plotting the respective coefficient of lift from the Python code, along with the analytical Thin Airfoil Theory, we see the following.

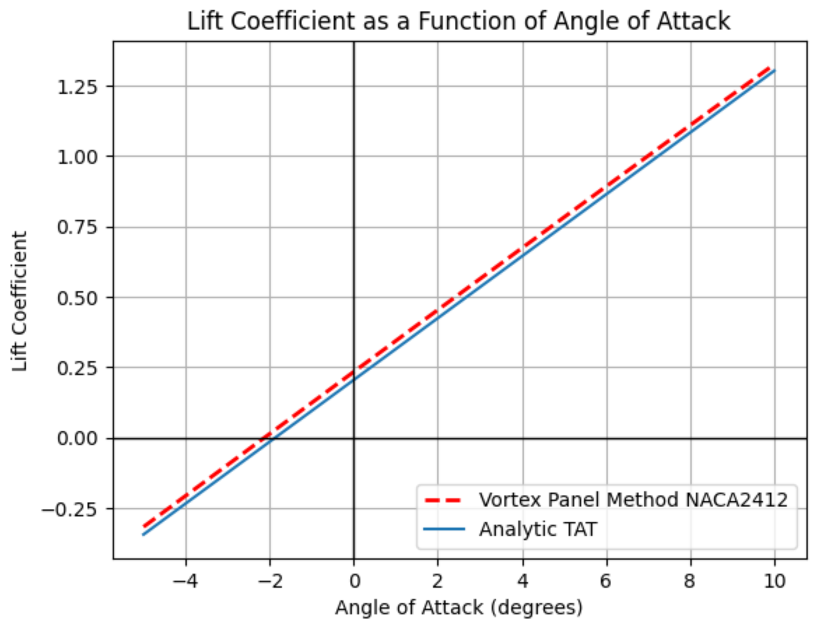

Fig. 19. Plot Comparison of Analytical Thin Airfoil Theory Calculation vs Vortex Panel Method

This graph above shows that our method for calculating the coefficient of lift using the vortex panel method lines up exactly with what we would expect from the analytic thin airfoil theory calculations. Because we were able to implement Python code that depended on the 4-digit NACA code passed to the function, this Python code works for all 4-digit NACA airfoils. The Python code also grants the ability to control the number of panels we generate, allowing the user to scale up or down the resolution, should they see fit. This python code works well and can be applied to various applications for calculating airfoil C_L, assuming that they are at low angles of attack. Although calculating it experimentally using a program such as XFLR5 is useful, being able to obtain another point of reference, especially when using personally developed method, is something that allows for a deeper understanding of the theory, and creative ways to implement that theory when doing calculations.

Note: The Python code requires Appendix B and Appendix C to be run in the same environment, as Appendix C is dependent on Appendix B.

Appendix

Appendix A: Code for plotting Cl vs Alpha of NACA 2412, NACA 0012 and P2

```
import numpy as np
import matplotlib.pyplot as plt

alpha_0_1 = -0.0362546*1.5      # Second function
alpha_0_2 = -0.032807*2.5  # Third function

# Define alpha range (x-axis values) in radians
alpha = np.linspace(-10, 10, 100)

# Calculate Cl for the three functions
Cl_main = 2 * np.pi * (alpha)
Cl_1 = 2 * np.pi * (alpha - alpha_0_1)
Cl_2 = 2 * np.pi * (alpha - alpha_0_2)

# Create the plot
plt.figure(figsize=(8, 6))

# Plot the three functions
plt.plot(alpha, Cl_main, label=r'$NACA-0012$', color='b', linewidth=2)
plt.plot(alpha, Cl_1, label=r'$NACA-2412$', color='r', linestyle='--', linewidth=2)
plt.plot(alpha, Cl_2, label=r'$P2-Individual$', color='g', linestyle='-.', linewidth=2)
plt.xlim(-0.75, 0.75)
plt.ylim(-3.6, 3.6)

tick_positions = np.linspace(-0.75, 0.75, 11)  # Tick positions in radians
tick_labels = np.degrees(tick_positions)   # Convert tick positions to degrees
plt.xticks(tick_positions, [f"{int(tick)}°" for tick in tick_labels])

plt.xlabel('Angle of Attack (α) [Degrees]', fontsize=12)
plt.ylabel('Lift Coefficient ($C_l$)', fontsize=12)
plt.title('Lift Coefficient vs Angle of Attack for NACA 0012, NACA 2412, and P2', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.show()
```

Appendix B: Code for plotting theoretical values of Cl vs alpha over experimental values

```
import numpy as np
import matplotlib.pyplot as plt

# Load the background image
background_image_path = 'EXPDATA.png'  # Replace with your actual image file path
img = plt.imread(background_image_path)  # Load the image as an array

plt.figure(figsize=(10,10))
plt.imshow(img,extent=[-32,32,-2,3.6],aspect = 17)
plt.autoscale(False)


# Define constants for the functions
alpha_0_1 = -0.0362546 # Second function
alpha_0_2 = -0.032807   # Third function

# Define alpha range (x-axis values) in radians
alpha = np.linspace(-32, 32, 100)

# Calculate Cl for the three functions
Cl_main = 2 * np.pi * np.radians(alpha)
Cl_1 = 2 * np.pi * (np.radians(alpha) - alpha_0_1)
Cl_2 = 2 * np.pi * (np.radians(alpha) - alpha_0_2)

plt.plot(alpha, Cl_1, label=r'$NACA-2412$', color='r', linestyle='--', linewidth=2, zorder=2)

#Add labels, title, grid, and legend
plt.xlabel('Angle of Attack (α) [Degrees]', fontsize=12)
plt.ylabel('Lift Coefficient ($C_l$)', fontsize=12)
plt.title('Lift Coefficient vs Angle of Attack for NACA 2412', fontsize=14)
plt.legend(fontsize=12)
```

Appendix C: Code for plotting lift coefficient vs angle of attack for NACA 2412 airfoil at varying Reynolds Numbers

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the NACA2412 data
file_name = "Polar_Graph.xlsx"
data = pd.read_excel(file_name)

# Convert columns to numeric values
alpha = pd.to_numeric(data['Alpha'], errors='coerce')
lift3 = pd.to_numeric(data['T1_Re3.000_M0.00_N9.0'], errors='coerce')
lift6 = pd.to_numeric(data['T1_Re6.000_M0.00_N9.0'], errors='coerce')
lift9 = pd.to_numeric(data['T1_Re9.000_M0.00_N9.0'], errors='coerce')

# Drop rows with NaN values
cleaned_data = pd.DataFrame({'Alpha': alpha, 'Lift3': lift3, 'Lift6': lift6, 'Lift9': lift9}).dropna()

# Extract data
alpha_cleaned = cleaned_data['Alpha']
lift3_cleaned = cleaned_data['Lift3']
lift6_cleaned = cleaned_data['Lift6']
lift9_cleaned = cleaned_data['Lift9']

# Create the plot
Code for plotting CD vs CL of XFLR5 data with different flap angles

file_path = 'CD vs CL flaps.csv'
data = pd.read_csv(file_path)
CL_40 = data.iloc[:, 0]
CD_40 = data.iloc[:, 1]
CL_20 = data.iloc[:, 3]
CD_20 = data.iloc[:, 4]
CL_10 = data.iloc[:, 6]
CD_10 = data.iloc[:, 7]
CL_0 = data.iloc[:, 9]
CD_0 = data.iloc[:, 10]
plt.figure(figsize=(10, 6))
plt.plot(alpha_cleaned, lift3_cleaned, label=r'Re = $3 \times 10^6$', linestyle='-')
plt.plot(alpha_cleaned, lift6_cleaned, label=r'Re = $6 \times 10^6$', linestyle='-')
plt.plot(alpha_cleaned, lift9_cleaned, label=r'Re = $9 \times 10^6$', linestyle='-')

# Set bounds for x-axis and ensure -8 and 12 are the bounds for our x-axis
plt.xlim(-8, 12)
plt.xticks(range(-8, 13, 2))  # Show ticks from -8 to 12 with step of 2

# Add labels, title, and legend
plt.xlabel(r'$\alpha$ (Angle of Attack in Degrees)')
plt.ylabel(r'Lift Coefficient ($C_l$)')
plt.title('Lift Coefficient vs Angle of Attack')
plt.plot(CL_40, CD_40, linestyle='-', color='r', label="40$^{\circ}$")
plt.plot(CL_20, CD_20, linestyle='-', color='y', label="20$^{\circ}$")
plt.plot(CL_10, CD_10, linestyle='-', color='g', label="10$^{\circ}$")
plt.plot(CL_0, CD_0, linestyle='-', color='b', label="0$^{\circ}$")
plt.xlabel('$C_L$')
plt.ylabel('$C_D$')
plt.title('$C_D$ vs $C_L$ Wing with 1/4 span flaps')
plt.legend()
plt.grid(True)

# Display the plot
plt.show()
```

Appendix D: Code for plotting CL vs alpha of XFLR5 data and theoretical data

```
file_path = 'CL vs alpha.csv'
data = pd.read_csv(file_path)
alpha = data.iloc[:, 0]
CL = data.iloc[:, 1]
elip = 4.71*(np.radians(alpha) + 0.03625)
taper = 4.361*(np.radians(alpha) + 0.03625)
straight = 4.204*(np.radians(alpha) + 0.03625)
plt.figure(figsize=(10, 6))
plt.plot(alpha, CL, linestyle='-', color='r', label="XFLR5 Data")
plt.plot(alpha, elip, label = "Eliptical Wing")
plt.plot(alpha, taper, label = "Tapered Wing")
plt.plot(alpha, straight, label = "Straight Wing")
plt.xlabel('$\\alpha$ (degrees)')
plt.ylabel('$C_L$')
plt.xlim(-8.5, 12.5)
plt.title('$C_L$ vs $\\alpha$')
plt.legend()
plt.grid(True)
plt.show()
```

Code for plotting CL vs CD of XFLR5 data and theoretical data

```
file_path = 'CL vs CD.csv'
data = pd.read_csv(file_path)
CD = data.iloc[:, 0]
CL = data.iloc[:, 1]
delta_taper = 0.01
delta_straight = 0.05
AR = 6
CDi_taper = CL**2 * (1 + delta_taper) / (np.pi * AR)
CDi_straight = CL**2 * (1 + delta_straight) / (np.pi * AR)
plt.figure(figsize=(10, 6))
plt.plot(CL, CD, linestyle='-', color='r', label="XFLR5 Data")
plt.plot(CL, CDi_taper, label = "Tapered Wing")
plt.plot(CL, CDi_taper, label = "Straight Wing")
plt.xlabel('$C_L$')
plt.ylabel('$C_D$')
plt.title('$C_D$ vs $C_L$')
plt.legend()
plt.grid(True)
plt.show()
```

Appendix E: Code for plotting CL vs alpha of XFLR5 data with different flap angles

```
file_path = 'CL vs alpha flaps.csv'
data = pd.read_csv(file_path)
alpha = data.iloc[:, 0]
CL_40 = data.iloc[:, 1]
CL_20 = data.iloc[:, 4]
CL_10 = data.iloc[:, 7]
CL_0 = data.iloc[:, 10]
plt.figure(figsize=(10, 6))
plt.plot(alpha, CL_40, linestyle='-', color='r', label="40$^{\circ}$")
plt.plot(alpha, CL_20, linestyle='-', color='y', label="20$^{\circ}$")
plt.plot(alpha, CL_10, linestyle='-', color='g', label="10$^{\circ}$")
plt.plot(alpha, CL_0, linestyle='-', color='b', label="0$^{\circ}$")
plt.xlabel('$\\alpha$ (degrees)')
plt.ylabel('$C_L$')
plt.xlim(-8.5, 12.5)
plt.title('$C_L$ vs $\\alpha$ with 1/4 span flaps ')
plt.legend()
plt.grid(True)
plt.show()
#Set Airfoil Coordinates
x_upper = x
x_lower = x
y_upper = y_upper
y_lower = y_lower
ycont = np.concatenate((y_upper[::-1], y_lower[1:n_points]))
xcont = np.concatenate((x_upper[::-1], x_lower[1:n_points]))
```

Appendix F: Code for Plotting NACA 4-Digit Airfoil

```
import numpy as np
import matplotlib.pyplot as plt

def naca4_digit_airfoil(code, n_points=100):
    chord_length=1.0

    m = int(code[0]) / 100.0  # Maximum camber
    p = int(code[1]) / 10.0   # Position of maximum camber (x/c)
    t = int(code[2:]) / 100.0 # Maximum thickness

    beta = np.linspace(0, np.pi, n_points)
    x = (1 - np.cos(beta)) / 2 * chord_length

    # Thickness distribution
    yt = (t / 0.2) * (
        0.2969 * np.sqrt(x / chord_length)
        - 0.1260 * (x / chord_length)
        - 0.3516 * (x / chord_length)**2
        + 0.2843 * (x / chord_length)**3
        - 0.1015 * (x / chord_length)**4
    )

    # Camber line and slope
    yc = np.zeros_like(x)
    dyc_dx = np.zeros_like(x)
    for i in range(len(x)):
        if x[i] < p * chord_length:
            yc[i] = (m / p**2) * (2 * p * (x[i] / chord_length) - (x[i] / chord_length)**2)
            dyc_dx[i] = (2 * m / p**2) * (p - x[i] / chord_length)
        else:
            yc[i] = (m / (1 - p)**2) * ((1 - 2 * p) + 2 * p * (x[i] / chord_length) - (x[i] / chord_length)**2)
            dyc_dx[i] = (2 * m / (1 - p)**2) * (p - x[i] / chord_length)

    theta = np.arctan(dyc_dx)

    # Upper and lower surfaces
    y_upper = yc + yt * np.cos(theta)
    y_lower = yc - yt * np.cos(theta)

    return x, y_upper, y_lower, yc,n_points

#Usage
naca_code = "2412" #Input Desired Airfoil
x, y_upper, y_lower, yc,n_points = naca4_digit_airfoil(naca_code)

# Plotting
plt.figure(figsize=(10, 5))
plt.plot(x, y_upper, label="Upper Surface", color="blue")
plt.plot(x, y_lower, label="Lower Surface", color="red")
plt.plot(x, yc, label="Camber Line", color="green", linestyle="--")
plt.fill_between(x, y_lower, y_upper, color="grey", alpha=0.5)
plt.title(f"NACA {naca_code} Airfoil")
plt.xlabel("x")
plt.ylabel("y")
plt.axis("equal")
```

Appendix G: Code for Calculating C_l Using Vortex Sheet

```
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy.linalg import solve

alpha_vals = np.linspace(-5,10,40, dtype=np.intc)
Cl_vals = np.ones(alpha_vals.shape[0])
Cm_vals = np.ones(alpha_vals.shape[0])
num_panels = 2*(x_upper.shape[0]-1)-1


for i in range(alpha_vals.shape[0]):
  alpha = alpha_vals[i]
  alpha_rad = alpha/180 * np.pi;

  # coordinates of the vorticies
  xv_upper = x_upper[:-1] + 0.25 * (x_upper[1:] - x_upper[:-1])
  yv_upper = y_upper[:-1] + 0.25 * (y_upper[1:] - y_upper[:-1])
  # coordinates of the control points
  xc_upper = x_upper[:-1] + 0.75 * (x_upper[1:] - x_upper[:-1])
  yc_upper = y_upper[:-1] + 0.75 * (y_upper[1:] - y_upper[:-1])
  # coordinates of the vorticies
  xv_lower = x_lower[:-1] + 0.25 * (x_lower[1:] - x_lower[:-1])
  yv_lower = y_lower[:-1] + 0.25 * (y_lower[1:] - y_lower[:-1])
  # coordinates of the control points
  xc_lower = x_lower[:-1] + 0.75 * (x_lower[1:] - x_lower[:-1])
  yc_lower = y_lower[:-1] + 0.75 * (y_lower[1:] - y_lower[:-1])

  # make the two surfaces into one array again
  xv = np.concatenate((xv_upper[::-1], xv_lower))
  xc = np.concatenate((xc_upper[::-1], xc_lower))
  yv = np.concatenate((yv_upper[::-1], yv_lower))
  yc = np.concatenate((yc_upper[::-1], yc_lower))

  A = np.ones((num_panels, num_panels))
  B = np.ones((num_panels,1))

  for p in range(num_panels):
    for q in range(num_panels):
      dxp = xcont[p+1]-xcont[p]
      dyp = ycont[p+1]-ycont[p]
      L = np.sqrt((dxp**2)+(dyp**2))


      R = np.sqrt(((xv[q] - xc[p]) ** 2 + (yv[q] - yc[p]) ** 2))
      cos_d2pq = (xv[q] - xc[p]) / R
      sin_d2pq = (yv[q] - yc[p]) / R

      cos_thetap = dxp / L
      sin_thetap = dyp / L

      numerator = cos_d2pq*cos_thetap + sin_d2pq*sin_thetap
      denominator = 2*np.pi*R
      A[p,q] = numerator/denominator

    thetap = np.arctan2(ycont[p+1] - ycont[p], xcont[p+1] - xcont[p])

    B[p] = np.sin(thetap - alpha_rad)

  Gamma = solve(A,B)
  Cl = 2*np.sum(Gamma)
  Cl_vals[i] = Cl

plt.figure(1)
plt.plot(alpha_vals,Cl_vals,'r--', linewidth = 2,label=f'Vortex Panel Method NACA{naca_code}')
plt.plot(alpha_vals, 2*np.pi*((alpha_vals*np.pi/180) + 0.0326),label='Analytic TAT')
plt.title('Lift Coefficient as a Function of Angle of Attack')
plt.xlabel('Angle of Attack (degrees)')
plt.ylabel('Lift Coefficient')
plt.legend()
plt.grid()
plt.axhline(0,linewidth = 1, color = 'black')
plt.axvline(0,linewidth = 1, color = 'black')
plt.show()
```