SA1 Final Report

Freddie Ancliff (frja2)

June 2024

1 Summary

Aerofoil design makes up a significant portion of the development of all aircraft both in terms of performance that can be extracted but also in terms of the time and computational resources required to tackle this task. Traditional CFD solutions of a 3D aerofoil with fuselage can take many hours to run, and wind tunnel testing can take days or even weeks when accounting for the high-tolerance manufacturing of test-parts.

It is thus desirable to able to analyse many different potential aerofoil designs quickly and efficiently but maintaining significant accuracy to draw meaningful conclusions during the design process. This report details such a technique using an inviscid lift solution with a panel method approximation, coupled with an empirical-based boundary layer growth model to solve for the drag of the aerofoil. The model is then used to analyse many 2D aerofoil sections and find optimum designs which maximise the lift/drag ratio of the aerofoil section for different flight conditions such as Reynolds number and incidence.

2 Introduction

As explained in the task handout [3], the key aim in aerofoil design is to optimise the Lift/Drag ratio for a given set of flight conditions. The reason for this is that the Specific Fuel Consumption, proportional to the fuel consumed per unit distance travelled, is directly proportional to the L/D ratio of the aircraft. Reducing the Specific Fuel Consumption by increasing L/D therefore reduces the cost of aircraft operations and also reduces the environmental impact by reducing the fuel burnt per journey. If we do not know the fuselage drag, which will be primarily due to skin friction as the fuselage is a very high aspect ratio body, the only way to optimise the aircraft L/D is to optimise the aerofoil L/D which is directly equivalent to optimising the aerofoil C_l/C_d ratio.

Despite not knowing the fuselage drag, we can assume that it is approximately constant over the small range of aircraft incidences tested in this analysis and therefore the optimum C_l/C_d ratio will be reduced but will remain at a similar angle of attack and be slightly shifted to higher angles of attack (higher absolute aerofoil lift and drag values thus less effect from constant fuselage drag) but not into a region causing significant separation of the aerofoil.

In this report we aim to test a simple (and therefore fast) numerical solver on different aerofoil sections and design 2 aerofoils with as high a C_l/C_d ratio as we can achieve, for 2 different Reynolds numbers (0.5e6 and 20e6). In practice when designing aerofoils for aircraft applications, we would aim to design our aerofoil to have the optimum C_l/C_d ratio at the intended cruise condition Reynolds number and incidence with some leeway either side of this optimum so that the aerofoil is stable during in-air manoeuvres, take-off and landing, and wind gusts or other forms of inlet turbulence and unsteadiness.

3 Numerical Model

The following section details the key working principles of the numerical model developed to analyse our 2D aerofoil section designs and calculate the inviscid lift as well as the viscous drag. We will follow the structure listed in the project handout provided [3] during the development of the model.

3.1 Lift

To calculate the lift of our 2D aerofoil sections we employ an inviscid panel method solution. In doing so, we calculate the circulation at each end of all the discrete panels by solving the matrix of equations relating the boundary of the aerofoil, the Kutta Condition at the Trailing Edge (TE), and the potential flow solution around a line vortex of linear strength variation between two specified end values and locations. In doing so we obtain the circulation at a set number of points along the aerofoil surface and the local flow velocity along the surface of the aerofoil is simply this circulation value (multiplied by -1 for the lower surface velocities). The local flow velocity is then related to Cp. The lift coefficient of the aerofoil is obtained by finding the total circulation of all panels around the aerofoil and then multiplying by -2 (as we are interested in the coefficient not the absolute value).

3.2 Drag

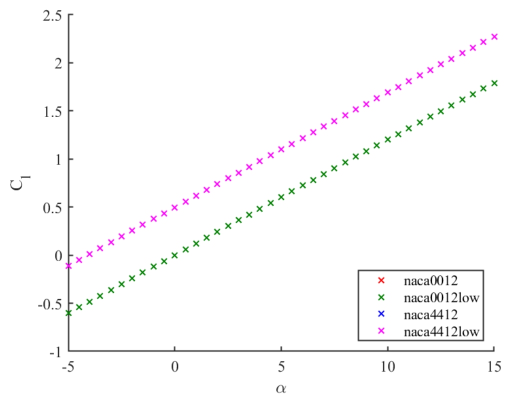
After calculating the Cp distribution around the aerofoil by relating to calculated surface velocities via Bernoulli's Equation, we can model the growth and behaviour of the boundary layer at each location along the aerofoil. As we are interested in finding the drag coefficient of the aerofoil we aim to eventually obtain the momentum thickness of both the upper and lower surface boundary layer at the TE of the aerofoil. This is done using an integral method as explained in the handout [3]. Starting at the stagnation point of the aerofoil the boundary layer is assumed to be laminar with 0 thickness. Growth of the boundary layer is then modelled moving from the stagnation point to the TE for both the upper and lower surfaces. Approximations to the transition and separation behaviour of the boundary layer are made using empirical relations (from Eppler and Somers [1]) relating the momentum, energy, and displacement thicknesses, to the surface Cp distribution as well as the Reynolds number of the free-stream.

Because of the empirical nature of this approximation the final C_d results will have some degree of inaccuracy but the effect of changing the Cp distribution or the Reynolds number will be well captured but absolute values may be inaccurate.

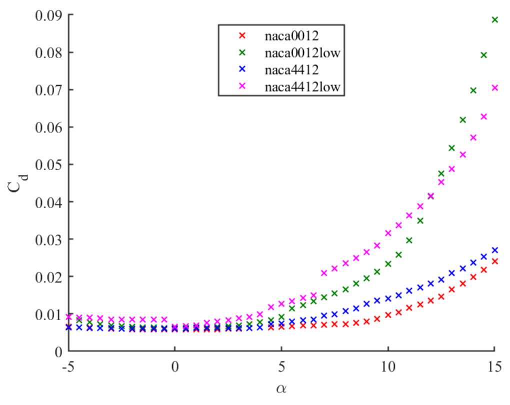
The boundary layer integral solver is able to account for natural transition, laminar separation, turbulent reattachment, and turbulent separation, the criteria for each again being taken from Eppler and Somers [1].

3.3 Validity and Accuracy

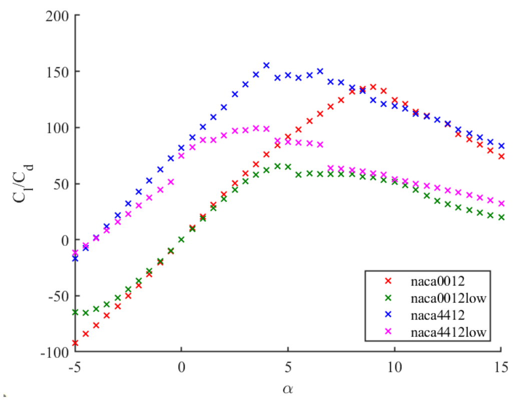
In order to assess the accuracy and performance of the model it was first tested on two NACA aerofoils (0012 and 4412) at a range of Reynolds Numbers. Figure 1 shows the characteristics of these two aerofoil sections at a Reynolds number of 20e6 and 5e6.



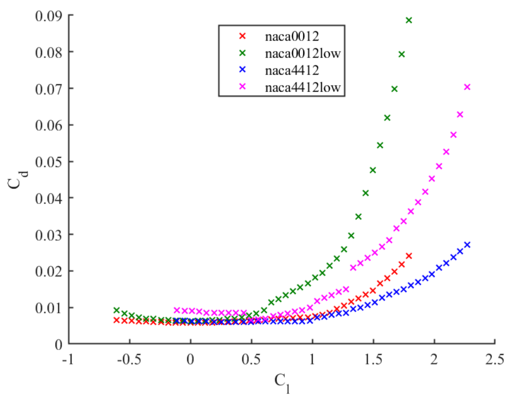
(a) C_l against α



(b) C_d against α



(c) C_l/C_d against α



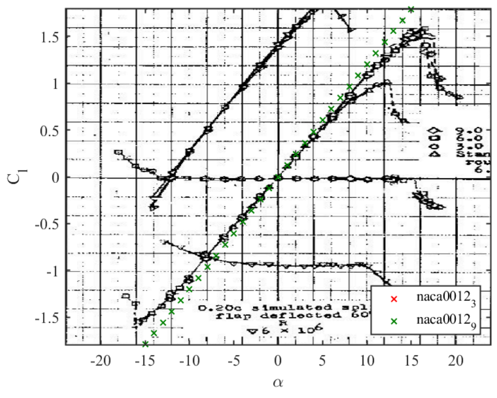
(d) C_d against C_l

Figure 1: Numerical Calculations for NACA0012 and 4412 at $Re = 20 * 10^6$, and $0.5 * 10^6$

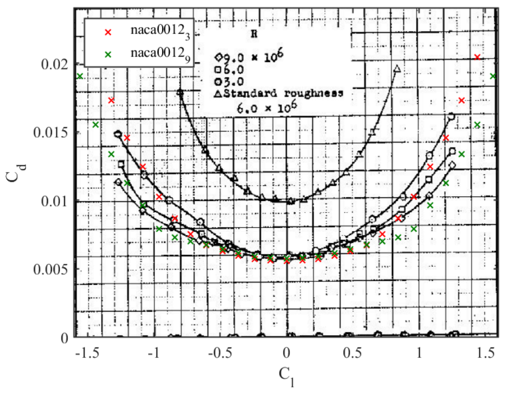
The behaviour of the 2 elements of the numerical model can clearly be observed in Figures 1a and 1b.

Figure 1a shows that the inviscid lift solution is not affected by changing Reynolds number for a given aerofoil section. This may present as trivial to the reader as the Reynolds number is dependent on viscosity so clearly an inviscid solution will be independent of the Reynolds number. Increasing the camber of the aerofoil from NACA0012 to NACA4412 results in a simple vertical offset of the C_l against α curve.

Figure 1b can be used to infer elements of the behaviour of the viscous boundary layer solver. Observing the pink curve of the NACA4412 at low Re, we notice a decrease in drag when α is increased above 0 degrees. At incidences below 0 we observe that laminar separation and turbulent reattachment occur at approximately 10% of the aerofoil chord, whereas at 0^o we observe no laminar separation instead the lower surface boundary layer undergoes natural transition at approximately 85% of the chord length. As such the lower surface boundary layer is laminar for more of the aerofoil chord and so the boundary layer thickness will grow at a lower rate and the viscous drag calculated will be lower. Studying the C_l/C_d against α curve for the NACA4412 at high Re (Figure 1c reveals a discrete drop in the C_l/C_d at an incidence of 5.5^o. No drop is observed in the C_l against α plot meaning the drop in lift/drag ratio must be due to a discrete increase in drag at this incidence. Analysis of the output from the boundary layer solver reveals that from $\alpha = 5^o$ to 5.5^o the location of turbulent separation moves from x = 1.000 (exactly at the TE) to x = 0.998 (slightly on the aerofoil surface). When turbulent separation occurs at x = 1.000 there is no effect on the calculated aerofoil drag as the boundary layer thickness at the trailing edge is not influenced by this separation. However, when separation moves marginally onto the aerofoil surface (from the TE point to the next discrete point) the boundary layer thickness at the TE and thus the calculated viscous drag is increased. This is why we see discrete jumps in the characteristics of Figure 1. The magnitude of these jumps can be reduced by decreasing the spacing between the points at which the boundary layer solver evaluates separation and transition criteria. This is done by increasing the number of panels by which we approximate the aerofoil surface and is discussed further in 3.3.1.

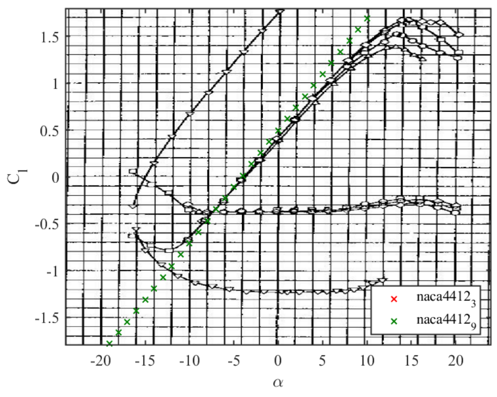


(a) C_l against α

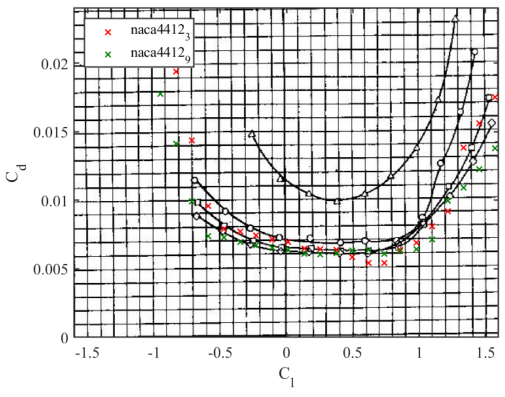


(b) C_d against C_l

Figure 2: Experimental vs Numerical Calculations for NACA0012 at $Re = 3x10*6$, and $9*10^6$



(a) C_l against α



(b) C_d against C_l

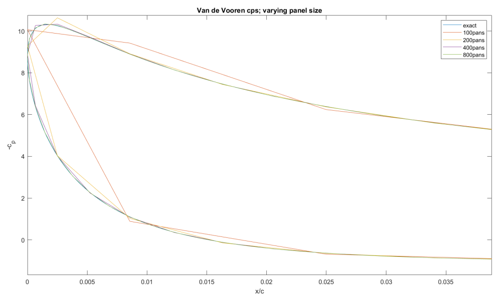
Figure 3: Experimental vs Numerical Calculations for NACA4412 at $Re = 3*10^6$, and $9*10^6$

By overlaying the calculated aerofoil characteristics with the experimentally generated plots for NACA0012 and 4412 from the handout [3], it is possible to analyse the validity of the model compared to the real-world behaviour. As seen from Figures 2a and 3a, the inviscid lift solution is accurate at low incidences where viscous effects do not significantly influence the flow around the aerofoil. As the incidence increases and turbulent separation begins to develop at the TE, our inviscid lift solution is unchanged as expected. The true viscous flow, however, will no longer exactly follow the aerofoil surface (as assumed by our inviscid solution) when turbulent separation occurs and the curvature of the flow will be reduced therefore altering the surface Cp distribution and reducing the magnitude of the expected lift. This is clearly seen in both Figures 2a and 3a when the true C_l against α curve peels away from the inviscid solution and is lower in magnitude at high magnitude incidences.

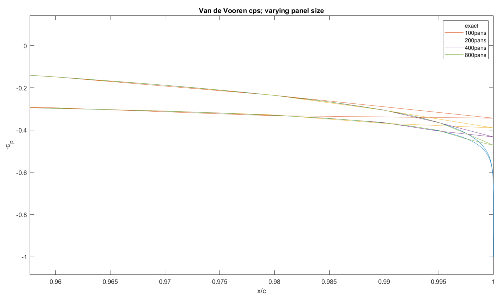
On the contrary, despite small inaccuracies in the C_l calculation at high angles of attack, the calculated C_d against C_l curves (for both NACA0012 and 4412, Figures 2b and 3b) overlay reasonably well with the experimental data for both Re = 3e6 and 9e6. At low angles of attack the numerical calculation curve is below the true curve and implies that the drag from our boundary layer solver is an underestimate at low incidences. Importantly for our work on designing aerofoils at different Reynolds numbers, the change in drag estimate from the boundary layer solution for the change of Re tested here shows the expected direction of change from the experimental results (even if the absolute value is not perfect).

It is worth noting that the boundary layer solver appears to perform better on the NACA0012 section (Figure 2) than on the NACA4412 section (Figure 3).

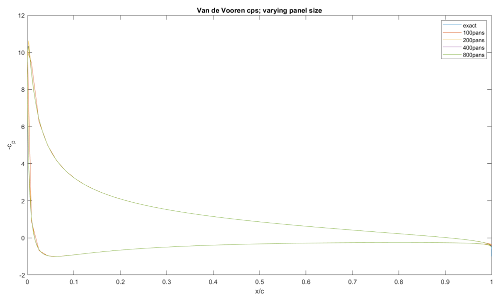
3.3.1 Increasing Model Accuracy



(a) Zoomed Cp Distribution at LE



(b) Zoomed Cp Distribution at TE



(c) Whole section Cp Distribution

Figure 4: Van de Vooren Aerofoil Cp Distribution calculated analytically and with various levels of discretisation using our numerical model

The accuracy of solutions from the inviscid lift panel method and viscous boundary layer solver can be improved by simply increasing the number of panels by which we approximate the aerofoil surface. This does, however, come at the cost of increased computation time for each condition (Re and α) solved. As the lift method relies on solving a 2D matrix, it is expected that the algorithmic complexity of the solution will be roughly $O(n^2)$. This was verified by testing the code for a sample Van de Vooren aerofoil and increasing the number of panles from 400 to 800 resulting in an increase in computation time per condition of approximately 3.5x from 0.25s to 0.9s on average.

One benefit of decreasing the panel size (by increasing number) is that the surface velocity and thus Cp is calculated at the end of each panel so reducing the distance between panel ends results in a smoother Cp distribution over the aerfoil as shown for the sample Van de Vooren aerofoil in Figure 4c.

A further benefit of increasing the resolution of points is demonstrated by Figure 1c. As discussed in 3.3, the discrete jumps in the C_d plot of each aerofoil often correspond to transition, or laminar/ turbulent separation locations moving from one point of the approximation to the next resulting in a step change in the boundary layer thickness at the TE. By decreasing the distance between points in the approximation, the transition/ separation location will jump a smaller distance between two different α values which will in turn decrease the magnitude of the step change in calculated viscous drag.

4 Design of an Optimum Aerofoil Section

4.1 Additional Design Tools

In order to design aerofoil sections both more efficiently and with smoother, desirable Cp distributions, 2 tools were added to the Wing Analysis Section Generator (WASG) tool provided [3]. These were a surface curvature analyser and plotter, and a reference geometry plotter. The `Matlab` scripts for these tools and the edited WASG script are included in the Appendices, 8. The functions were linked to 3 keyboard keys, v and x to plot the upper and lower surface curvature respectively, and m to show the specified reference geometry/s. Parameters were included in the WASG tool for the following: reducing the number of curvature vectors plotted (for efficiency and/or visual reasons), scaling of the curvature vectors to a desired size, and choosing whether to plot the vector curvature (i.e reflexes cross the aerofoil surface and are plotted internally) or the absolute curvature (all curvature lines plotted out from the aerofoil).

Figure 6 shows how the curvature analysis tool can be used to modify the Cp distribution in a specified manner by altering the surface curvature. In this case the curvature at approximately 10% of the chord has been increased which locally decreases the LE curvature before this point and reduces the curvature over the rest of the top surface of the aerofoil. The results of this change can clearly be seen in 6c. At the location where curvature is locally increased we see an increase in magnitude of Cp as expected and before and after this x location we see a decrease in Cp as expected also from a decrease in magnitude of the surface curvature. By using this curvature tool in both the whole aerofoil view and in zoomed mode (both update simultaneously so can be manipulated side-by-side as shown in Figure 5), it should be apparent how the tool allows for smoothing of an existing aerofoil with an undesirable Cp distribution and also for creating aerofoils from scratch with a desired new Cp distribution.

Figure 7 shows how the reference geometry plotter tool can be used during the aerofoil design process to compare to a selection of different previously designed aerofoil surfaces, and can also be used in conjunction with aerofoil designs from literature and other sources if the surface co-ordinates are known.

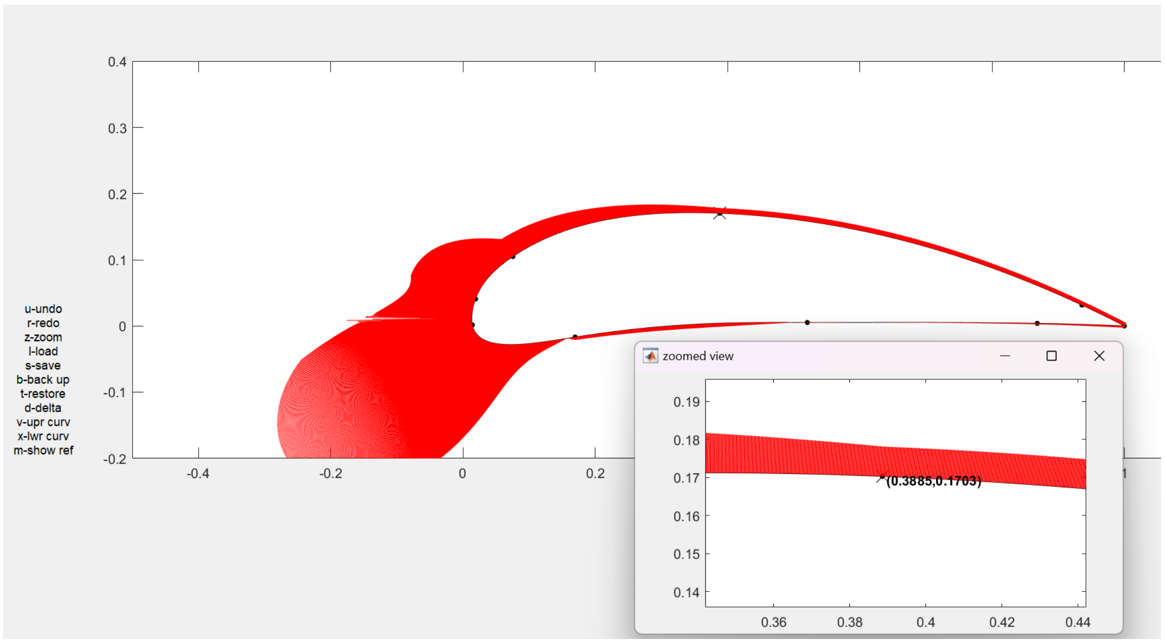
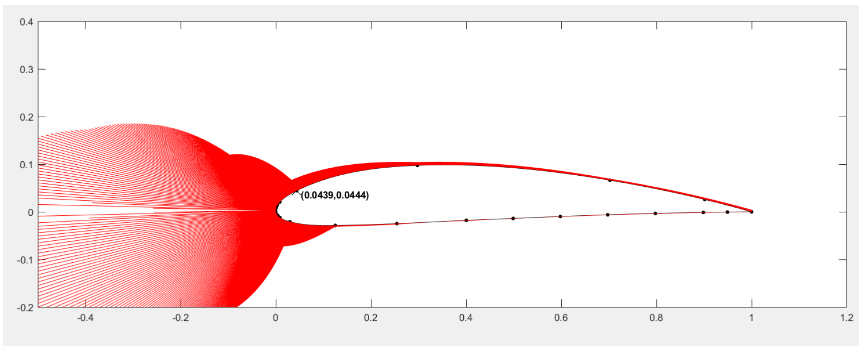
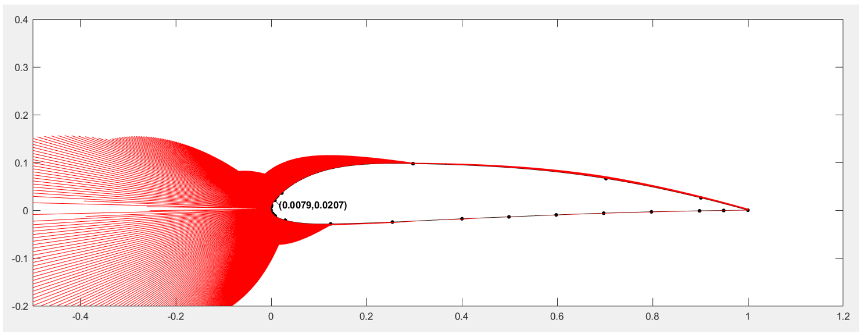


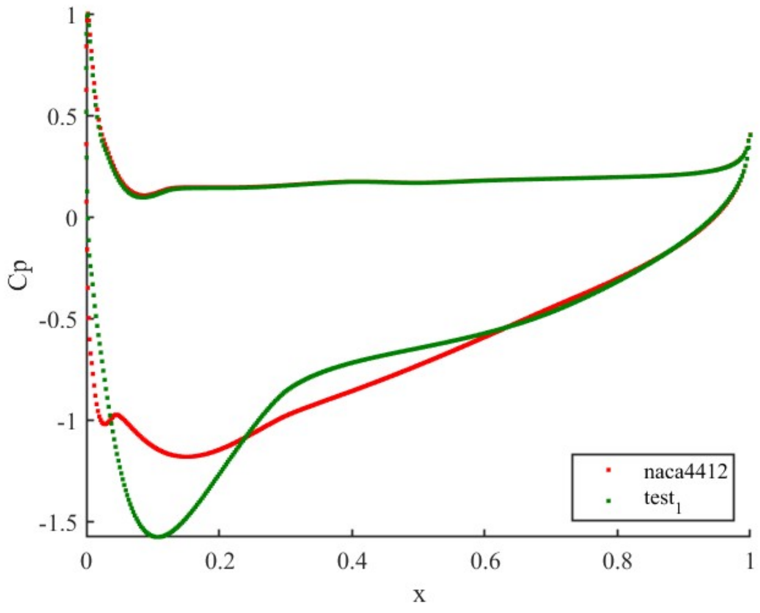
Figure 5: Curvature Tool Screenshot of Thick3 aerofoil design



(a) NACA4412



(b) NACA4412 with additional curvature close to leading edge and reduced curvature over the rest of the top surface



(c) C_p plot for both aerofoils

Figure 6: Example of curvature changing C_p plot

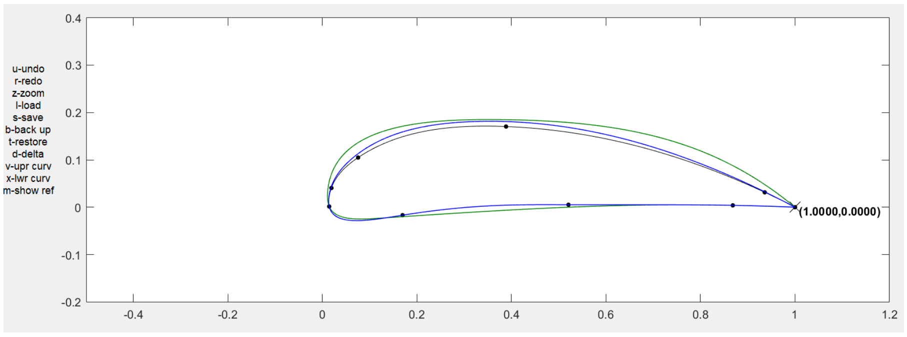


Figure 7: Reference Geometry Plotter Screenshot of Thick3 aerofoil being designed with the previous 2 iterations (Thick2 and Thick1 as reference geometries.

One improvement that could be made to the curvature analysis tool is the addition of a calculation of the derivative of the curvature to look for stationary points and label the curvature at this location to further aid development of as smooth a curvature distribution as possible. The lack of this feature, however, was not a significant effect at all on the development of any aerofoils during this work and is more a ”nice-to-have” than a major step in productivity.

The curvature analysis tool adds a 33ms time delay to the WASG tool for each mouse movement/ input when plotting both the lower and upper surface curvatures. If we assume this is the dominant computation time for each mouse movement then the WASG tool with curvature analysis will run at approximately 30 frames per second. In reality including other code items this will probably be closer to 15 or 20fps. These figures are entirely acceptable for a tool of this kind which is used for careful development and is not a game engine for example where fps performance is more critical to the end user.

The reference geometry plotting function has very little effect on code performance in terms of computation time. Trivially, plotting tens of reference geometries will have a more significant time penalty than plotting 1 geometry but it was found in our work that plotting any more than 4 reference geometries at one time was not helpful in any way and only obscured the actual surface in development. In addition to this, it is rarely necessary to plot more than 1 or 2 at any one time as the geometries plotted can be edited, changed, or removed whilst running the WASG tool. This is done by editing the `plot_ref_geom.m` function and saving this function then pressing ’m’ twice to hide the old geometries and then show the new reference geometries.

4.2 Intermediary Designs

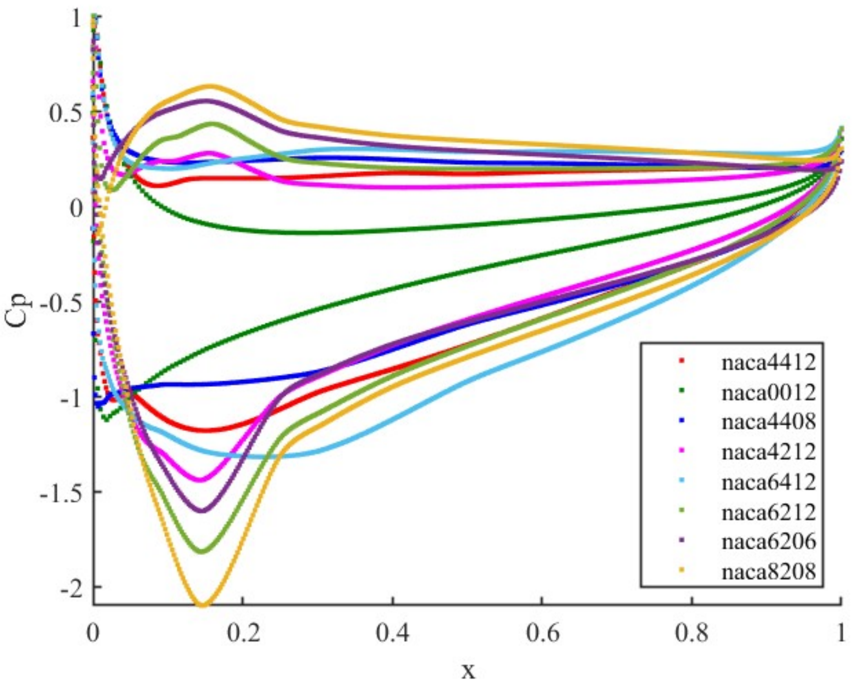
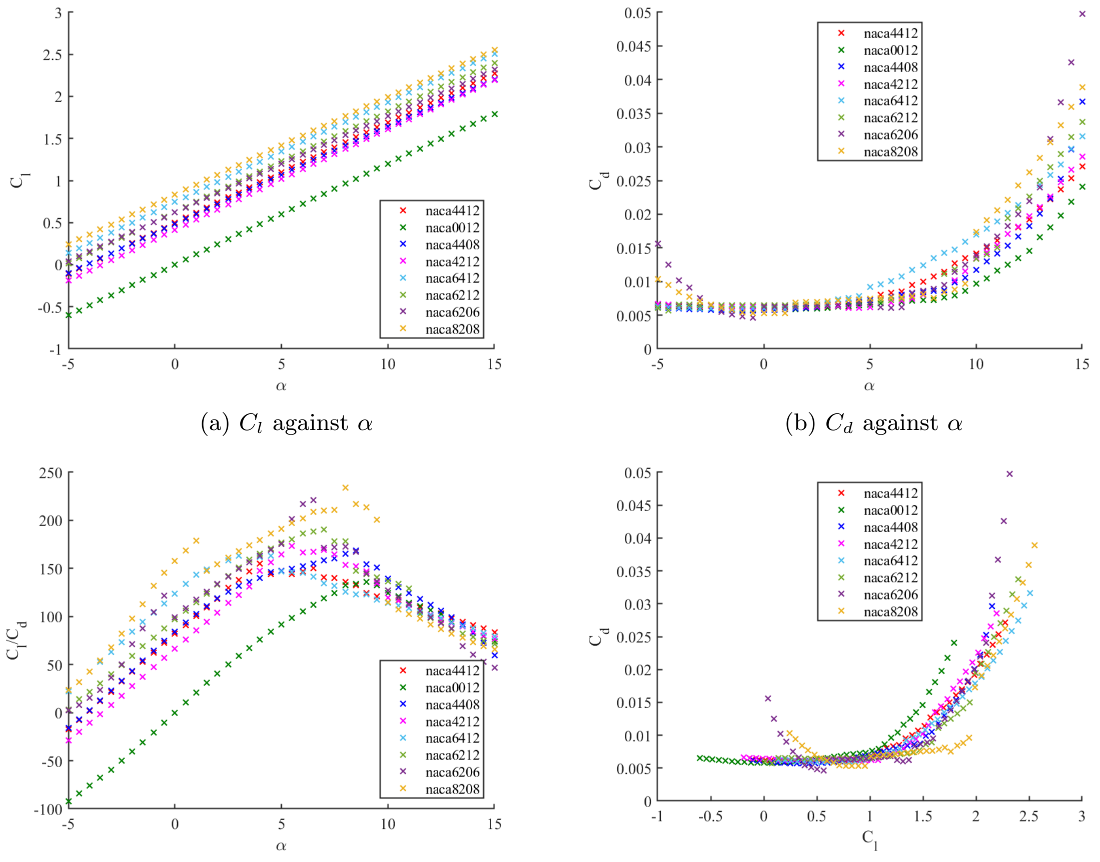


Figure 8: NACA Aerofoils at Re = 20e6, $\alpha = 3^o$



(a) C_l against α

(b) C_d against α

(c) C_l/C_d against α

(d) C_d against C_l

Figure 9: Characteristics of various NACA Aerofoils at $Re = 20 * 10^6$

NACA aerofoils make a good starting point for our section development as they are well researched and documented and the surfaces are easy to generate from an equation based on the 4 digit NACA code. A selection of NACA aerofoils and their characteristics are plotted above in Figures 8 and 9. It can be seen from the above figures that the 4-digit NACA aerofoil code allows for a wide variety of different aerofoil sections to be produced.

It is possible to study the effect of different section modifications on the aerofoil characteristics by comparing certain NACA aerofoils from these plots.

We can analyse the effect of increasing thickness by comparing the difference between NACA4408 (blue) and 4412 (red). The Cp distribution of the thicker aerofoil has a slightly higher magnitude peak at about 15% of the chord length whereas the suction peak of the lower thickness section is very close to the LE. Although the Cp distribution is only shown here for 1 incidence, the effect of thickness over multiple incidences can be seen in both Section 5.5 and Figure 9b. The results of 5.5 clearly show that there is a maximum thickness above which it is very hard to recover to the TE point without significant turbulent separation and the associated drag penalty. A cylinder would be one extreme of thickness distribution and clearly imposes a large viscous drag penalty. On the other hand, too little thickness can result in a very sharp suction peak around the LE at incidences that are not the design angle and thus can result in stall, particularly the very dangerous LE stall, at angles of attack which are not the design angle. The extreme at this end in terms of thickness would be a flat plate which clearly demonstrates significant LE separation at even very small incidences.

We can also gain an understanding into the effect of camber magnitude and camber location on the behaviour of our aerofoil sections through analysis of the tested NACA aerofoils. Comparing NACA0012 (green), 4412 (red), and 6412 (light blue) reveals that increasing the camber of the aerofoil both moves the suction peak backwards and increases its magnitude at a given incidence. As a result of this, the adverse pressure gradient (APG) behind the suction peak is more severe for higher cambered aerofoils which results in a higher tendency for turbulent separation at lower incidences than less cambered aerofoils. This effect is evident in Figure 9c where we can see that increasing camber increases the magnitude of C_l/C_d but that the peak of this ratio occurs at progressively lower incidences due to turbulent separation occurring earlier in the α range for more cambered aerofoils.

Finally, the effect of peak camber location can also be studied by comparing NACA4412 (red) against 4212 (pink), and NACA6412 (light blue) against 6212 (light green). It is evident from the Cp distribution 8, that in both cases moving the peak camber location forwards from 40% of chord to 20% locally increases curvature in this region whilst reducing curvature over the aft portion of the aerofoil. The result of this is a lower APG over the rear section of the aerofoil upper surface which improves resilience to turbulent separation at the TE. This is evident in the C_l/C_d against α curves 9c, where the 4212 and 6212 aerofoils both undergo turbulent separation at the TE (characterised by the discrete drop in the lift/drag ratio) at higher angles of attack than their 4412 and 6412 counterparts. This allows them to achieve higher lift/drag ratios as the lift coefficient keeps increasing linearly with α until turbulent separation occurs.

5 Optimum Aerofoil Section and Characteristics

5.1 High Reynolds Number - 20x10^6

For the design of the high Re aerofoil section, we started from a selected high performing NACA section. This was chosen to be NACA6212 which had a good balance of peak C_l/C_d without excessive suction peaks in the Cp distribution which were demonstrated by NACA8208 leading to earlier turbulent separation. We then tested modifications to the base NACA surface that were inspired by literature research, such as the Wortmann aerofoil [6] in Figure 10, and the BezierGAN generated aerofoil [2] in Figure 11. It was found that the lower surface reflex outperformed the upper surface reflex aerofoil despite surprisingly good performance from the upper surface reflex.

Upon arriving at an apparent optimum in the form of Figure 12a, small changes to the curvature distribution were made and refined using the curvature analysis tool 4.1, and during this time we also tested small changes around the optimum in terms of thickness and x-location of the peak camber of the aerofoil whilst comparing to the old geometry using the reference surface plotter 4.1, to ensure that the changes we made were not insignificant but also not too large a departure from the original design.

In doing so we discovered that moving the location of peak camber forward, Figure 12b, proved beneficial at increasing the range of incidences the aerofoil could operate at before turbulent separation occurs from 7 degrees to 9 degrees. As with studies of this on the NACA aerofoils, the resultant change in the Cp distribution was a sharper initial suction peak but a reduced pressure gradient over the rear portion of the aerofoil.

Both high Re aerofoils clearly outperformed NACA0012 and 4412 at the aim of optimising section C_l/C_d but at the cost of a larger peak in Cp distribution and therefore a greater risk of dangerous LE stall as well as a smaller operating range of incidences for which turbulent separation does not occur.

The C_l/C_d values achieved were 206 and 207 respectively.

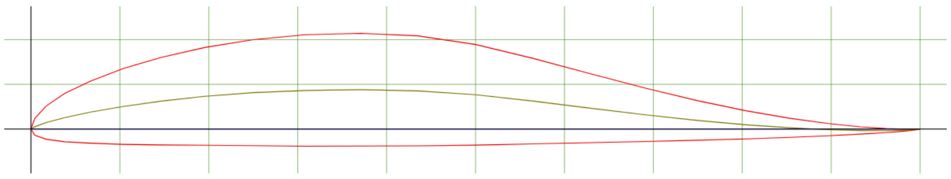


Figure 10: WORTMANN FX 05-H-126 aerofoil [6]

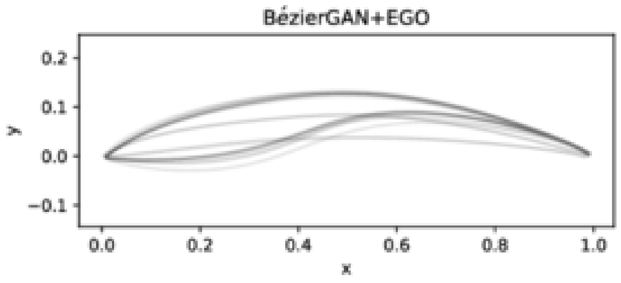
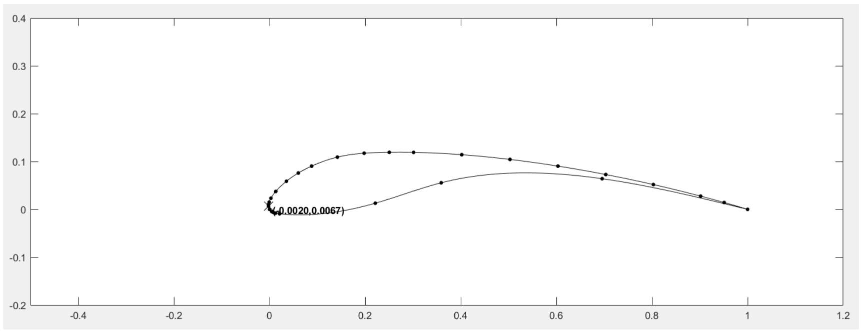
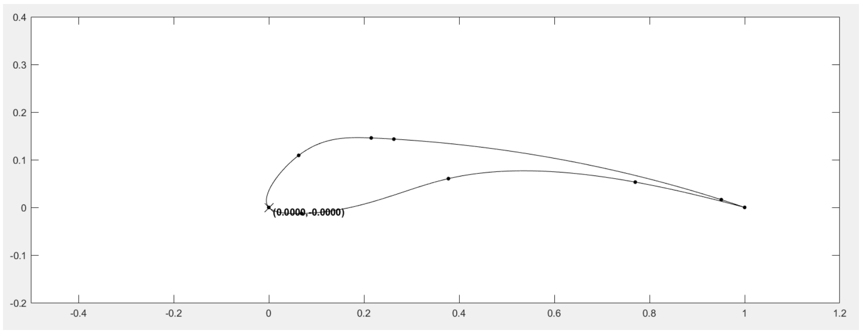


Figure 11: BezierGAN generated aerofoil [2]

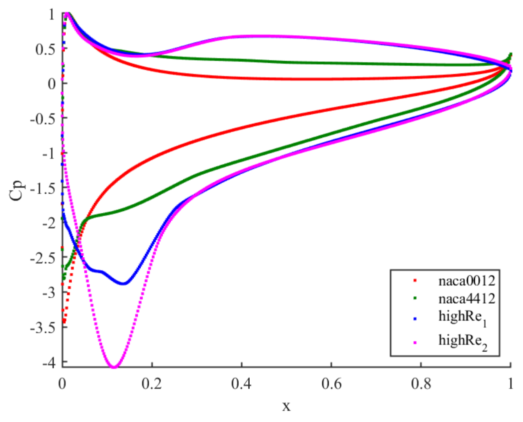


(a) $HighRe_1$ Aerofoil

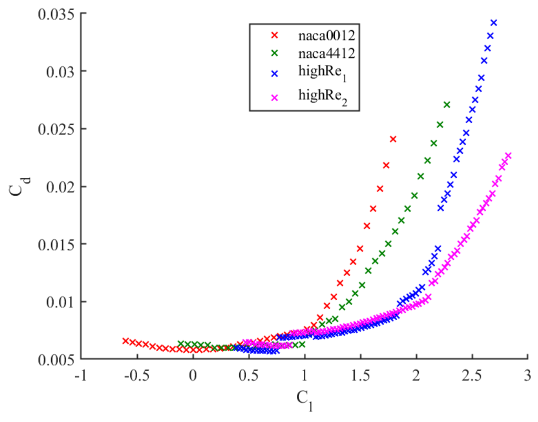


(b) $HighRe_2$ Aerofoil

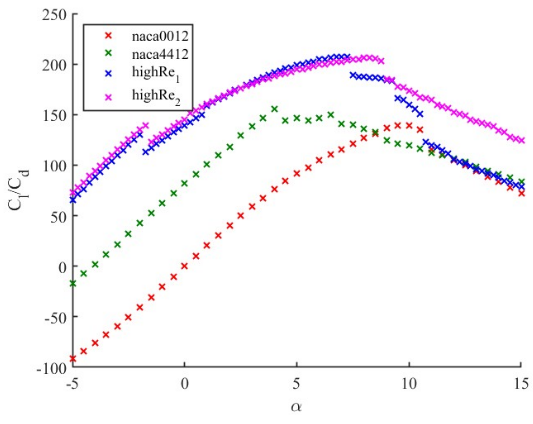
Figure 12: High Reynolds Aerofoils



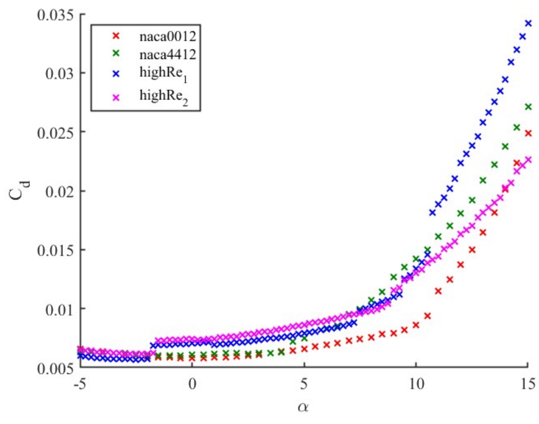
(a) C_p plot at $\alpha = 7^o$



(b) C_d against C_l plot



(c) C_l/C_d against α plot



(d) C_d against α plot

Figure 13: Characteristics of the high Reynolds aerofoil designs (Re = 20e6)

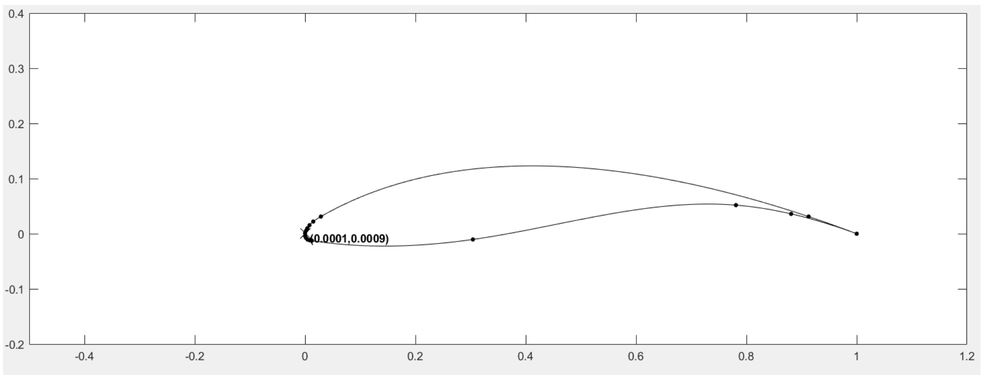
5.2 Low Reynolds Number - $0.5\text{x}10^6$

The design process for the low Re aerofoils followed a similar method to the high Re sections. In this case we started from our first high Re aerofoil 12a as we believed that the lower Re flow would not tolerate the significant suction peak of the second high Re aerofoil which we verified with a test. After testing both high Re aerofoils, we discovered the first performed the best but that both had significant separation issues around the strong suction peak as well as the extreme lower reflex due to the lower APG that could be tolerated by the boundary layer at the low Re number before separation occurred.

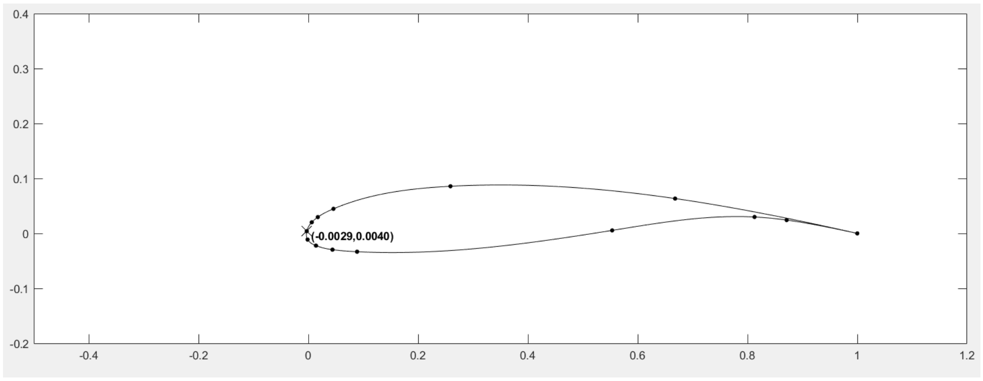
By reducing the curvature around the separated features of our fist high Re aerofoil at low Re we arrived at our first low Re aerofoil, Figure 14a. However, as can be seen from Figure 15, the aerofoil is only free from turbulent separation over a small range of angles (0 to 1.5 degrees), below 0 degrees the lower surface is separated and as such C_l/C_d measurements are inaccurate). This is a very small operating range and would make the aerofoil very unsuitable for most applications. Furthermore, at incidences past the optimum, LE separation of the upper surface occurs which leads to the very dangerous characteristic of LE stall resulting in sudden and large loss of lift and control because separation extends over the aerofoil control surfaces.

As such we aimed to design a second low Re aerofoil, arriving at Figure 14b. Care was taken to reduce the APG immediately after the suction peak and therefore delay laminar separation behind the suction peak which was experienced by the first low Re design for many incidences. To do this we reduced the curvature in this region using the curvature analysis tool and also reduced the curvature around the leading edge by thickening and rounding the LE to reduce lower and upper surface separation at the LE seen for incidences outside of the small operating range. As seen from the C_l/C_d against α curve in Figure 15, the changes had the desired effect on the Cp distribution and also increased the aerofoil operating range of incidences to -5 to 4 degrees before significant separation occurred. This came at the cost of peak C_l/C_d unfortunately which was not unexpected as we decambered the initial low Re section design.

The maximum C_l/C_d value for each aerofoil is 131.9 and 109 respectively.

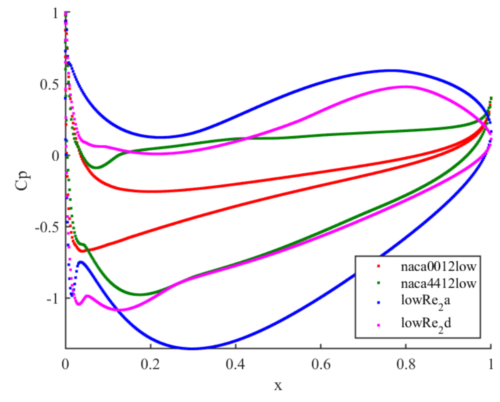


(a) $LowRe_2a$ Aerofoil

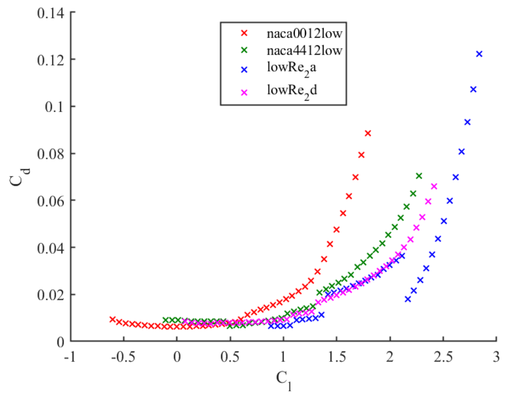


(b) $LowRe_2d$ Aerofoil

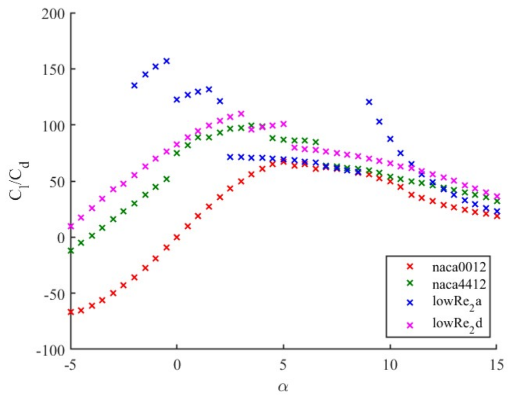
Figure 14: Low Reynolds Aerofoils (Re = 0.5e6)



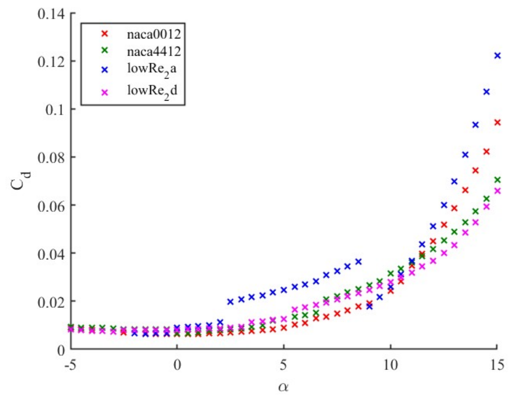
(a) C_p plot at $\alpha = 1.5^o$



(b) C_d against C_l plot



(c) C_l/C_d against α plot



(d) C_d against α plot

Figure 15: Characteristics of the low Reynolds aerofoil designs (Re = 0.5e6)

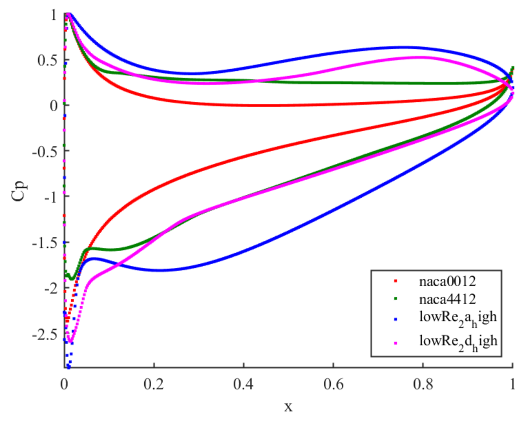
5.3 Comparison of High and Low Re sections

The reason for the very different behaviour of the high Re aerofoil sections when tested at low Reynolds number is because at low Re the boundary layer does not undergo natural transition before the suction peak as it does at the higher Reynolds number. Therefore, the boundary layer velocity profile is much less "full" and can therefore not sustain as severe an APG for as long as a turbulent boundary layer could before the wall-normal velocity gradient reaches 0 and separation occurs. In the case of the low Re aerofoils, laminar separation is often followed by turbulent reattachment shortly after but this carries with it a significant drag penalty and in extreme cases turbulent reattachment does not occur in which case the boundary layer is separated from around 20% of the chord length. As such when designing for low Re we must design Cp distributions that are both lower in magnitude and shallower to reduce APGs; this leads to aerofoils that are less cambered than their high Re counterparts.

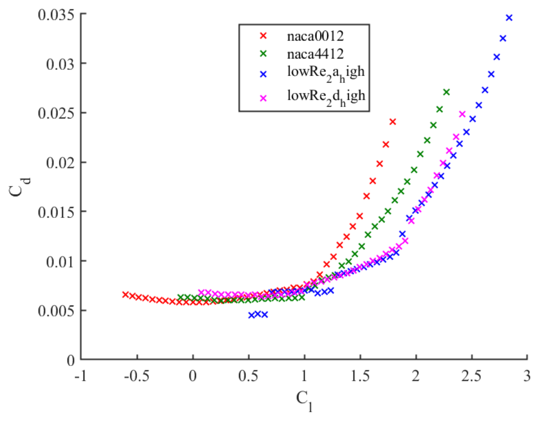
5.4 Designing for a Range of Reynolds Numbers

The reason we would want to design an aerofoil to perform well at a range of different Reynolds numbers and not have any dangerous stall characteristics such as LE stall is that this allows for various different flight speeds (Re dependent on V) and altitudes(Re dependent on ρ which reduces at altitude) to be available to the user depending on local weather conditions or route duration as well as many other factors. Furthermore, the aerofoil will also experience a range of Reynolds numbers not just at different cruise conditions but during take off and landing when it is imperative that any stall due to separation must be gradual and start from the trailing edge.

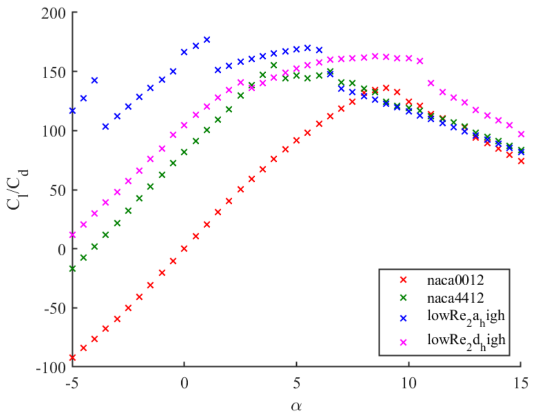
Figure 16 below shows the result of testing the two low Re aerofoil sections at a Reynolds number of 20e6. Both outperform the NACA0012 and 4412 aerofoils but not significantly and as at low Re the second design is operable over a wider range of incidences. Note that the operating range of both aerofoils is spread at higher Re as mentioned above due to natural transition before the suction peak on the upper surface.



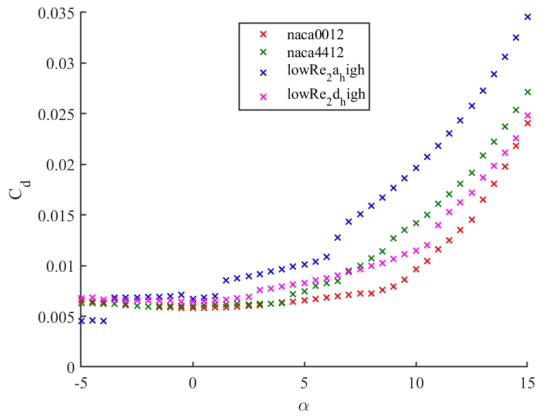
(a) C_p plot at $\alpha = 5.5^o$



(b) C_d against C_l plot



(c) C_l/C_d against α plot

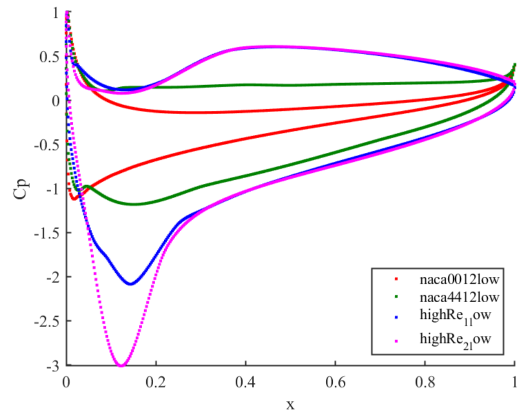


(d) C_d against α plot

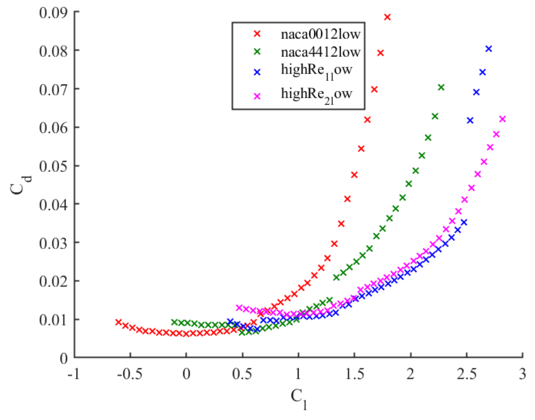
Figure 16: Characteristics of the low Re aerofoil designs tested at high Reynolds Number (Re = 20e6)

On the other hand, Figure 17 below demonstrates the effects mentioned in 5.2 that turbulent separation occurs at low incidences and as such the operating incidence range of both aerofoils is much smaller than at high Re. As we would expect the first high Re design outperforms the second at low Re in terms of C_l/C_d and delays turbulent separation by 1.5 degrees. Although neither aerofoil exhibits LE turbulent separation at this Re of 0.5e6, the severity of the suction peak of the second high Re design is unnerving and would not be recommended at all for any application involving flight at low Re due to the significant risk of dangerous LE stall.

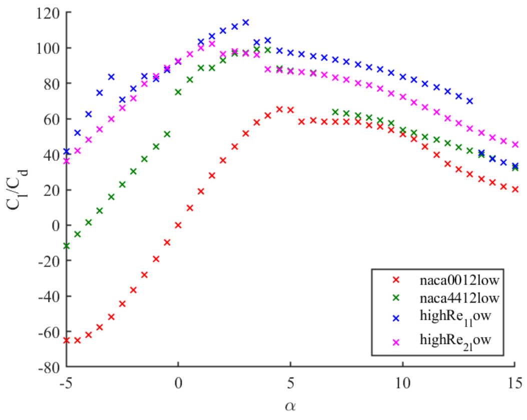
In summary for designing at multiple Reynolds numbers it would appear to be best to initially design at low Re due to the less tolerant boundary layers as demonstrated in this section. A smoother Cp distribution lends better to performance over a range of Reynolds numbers but sacrifices maximum C_l/C_d performance at higher Reynolds numbers when the boundary layer can remain attached over much more aggressive curvature geometries.



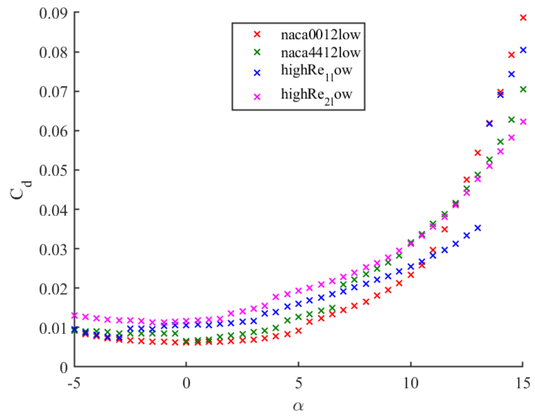
(a) C_p plot at $\alpha = 3^o$



(b) C_d against C_l plot



(c) C_l/C_d against α plot



(d) C_d against α plot

Figure 17: Characteristics of the high Re aerofoil designs tested at low Reynolds Number (Re = 0.5e6)

5.5 Designing for a Range of Incidences

It is often desirable to design an aerofoil that can operate over a fairly large range of incidences without significant compromise to the section performance or stall characteristics. This is the case to some extent for all aircraft aerofoils due to the unpredictability of wind gusts and the desire to avoid significant separation due to an unsteady wind changing the incidence of the aerofoil rapidly. The key application requiring good performance over a very large range of incidences is for cargo carrying aircraft as they will need to alter the wing incidence for a given flight speed to adjust for the weight of cargo being transported.

The key way to maximise the performance of the aerofoil over a large range of incidences is to reduce and large peaks in the pressure gradient at the leading edge over a wide range of incidences. This is done by increasing the thickness and therefore reducing the local curvature around the LE. The effect of this can be seen in Figures 5 and 6 between the thick3 aerofoil design and a NACA4412 using the curvature analysis tool.

As with most design choices, a compromise must be made on the thickness. Excessive LE thickness leads to too much APG to overcome at the TE and therefore TE separation. Lack of LE thickness leads to dangerous LE separation at high angles of attack. The two extremes can be thought of as a cylinder which has severe TE separation at all angles of attack, and a flat plate which has severe LE separation at incidences a small distance from 0^o

3 example thick aerofoils are shown in Figure 18. The Cp distributions at various incidences as well as the characteristic curves are shown in Figures 19 and 20 respectively. The characteristic curves for C_l/C_d are an overestimate of the true performance of these aerofoils as all 3 sections exhibit some degree of TE turbulent separation which is reduced in severity from thick1 to 2 to 3. The effect of reducing thickness from 1 to 2 to 3 serves to heighten the peak of the C_l/C_d curve whilst beginning to pay the price of higher drag at incidences far from the optimum.

Clearly the 3 sections shown are excessively thick as they result in TE separation at all incidences even at Re = 20e6. However, the effect of such large thickness on the Cp distribution 19 at various incidences is very interesting to compare against NACA0012 and 4412 (comparatively very thin aerofoils). It can be seen, especially for thick3, that the Cp distribution over a 20^o range of angle of attack is remarkably smooth on the upper surface. This can be compared to the behaviour of the much thinner aerofoils of NACA0012 and 4412 which exhibit severe peaks in the pressure distributions at incidences far from their design optimum leading to turbulent TE separation and a smaller operating range than the thicker aerofoils.

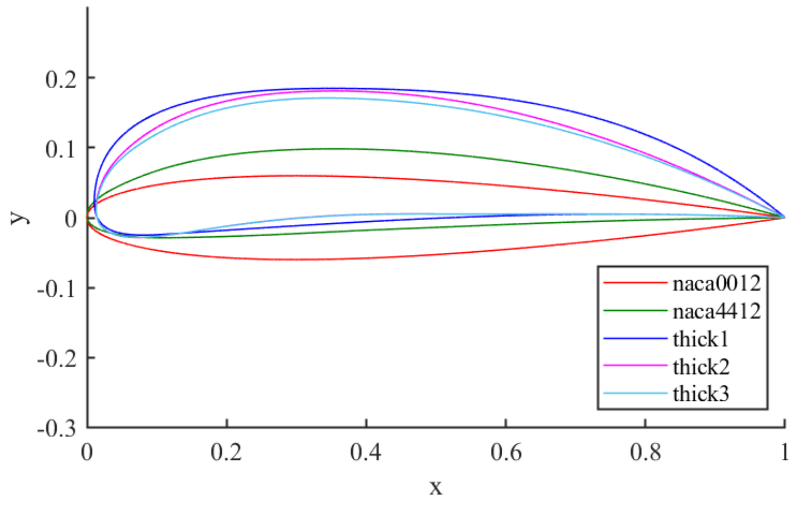
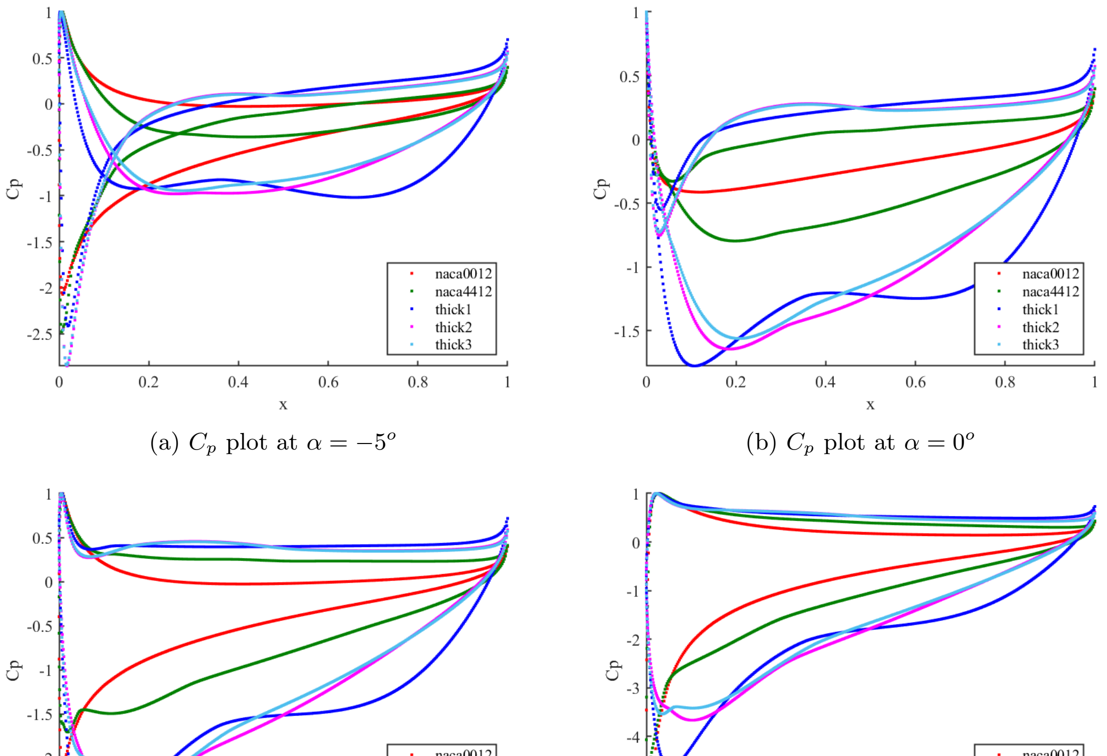


Figure 18: Surface Geometries of 3 Thick Aerofoils



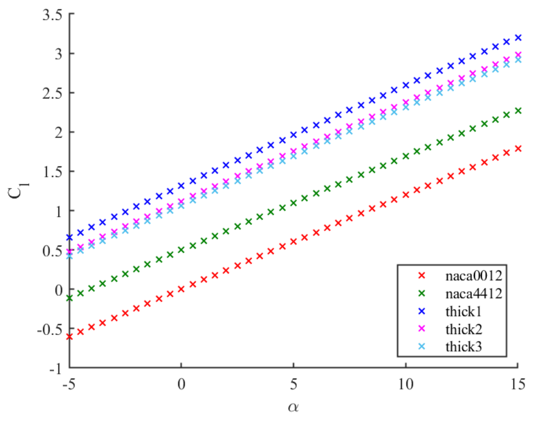
(a) C_p plot at $\alpha = -5^o$

(b) C_p plot at $\alpha = 0^o$

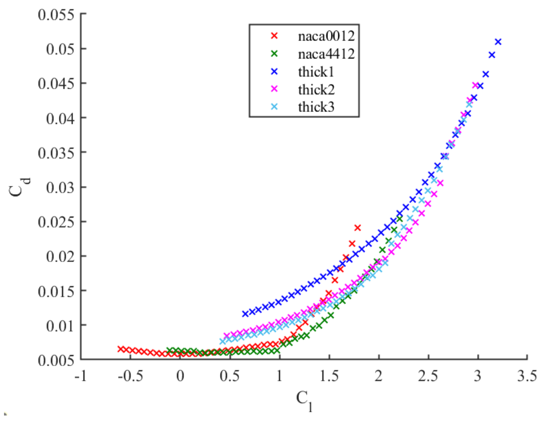
(c) C_p plot at $\alpha = 5^o$

(d) C_p plot at $\alpha = 10^o$

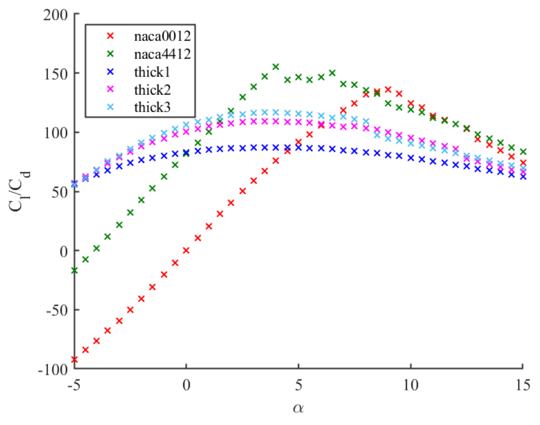
Figure 19: Surface Cp distribution of the 3 thick aerofoils tested over a range of incidences



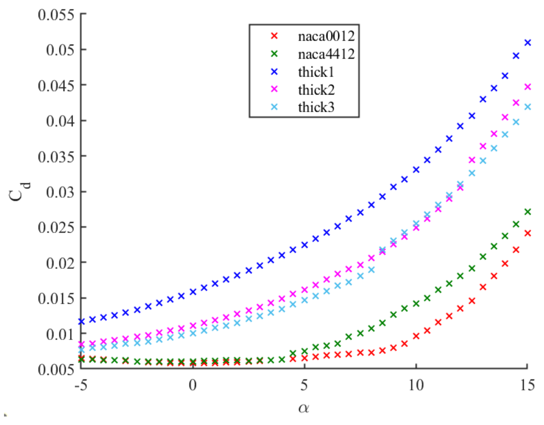
(a) C_l against α plot



(b) C_d against C_l plot



(c) C_l/C_d against α plot



(d) C_d against α plot

Figure 20: Characteristics of the 3 thick aerofoil surfaces tested targeting performance over a range of incidences

5.6 Real World Applicability of the Optimum Section Results

There are a variety of factors which have not been considered in this analysis but will impact the real world design and performance of aerofoil sections. A few examples and their possible effect on the optimum section results are included below.

1. Surface roughness of an aerofoil is an important factor to consider. For both the low and high Re aerofoils, surface roughness may prompt early boundary layer transition to turbulence. If transition to turbulence is initiated early due to surface roughness then the viscous drag will increase as a result of this (this is seen in Figures 2b and 3b. However, for any low Re aerofoils which suffer from LE separation around the suction peak, a roughness induced transition to a turbulent boundary layer before the suction peak will alleviate the risk of separation. This is because a turbulent boundary layer has a fuller velocity profile near the wall and thus can withstand higher Adverse Pressure Gradients (APGs) behind the suction peak before the velocity gradient at the wall reaches 0 and separation occurs. A similar effect to that of surface roughness on low Re aerofoils suffering from LE separation can be achieved using vortex generators to manually introduce turbulence into the boundary layer as well as thickening the boundary layer velocity profile near the wall through vortex induced mixing of high momentum free-stream flow with the near-wall flow.

2. The thickness of the TE is another factor that must be considered when designing real world aerofoils. For example, the very long and slender aft portions (past half chord) of the 2 high Re designs (Figure 12) are highly unlikely to be structurally feasible, especially when considering housing for various mechanical systems such as hydraulic flap actuators and landing gear storage.

3. Non-uniform and unsteady flow onto the aerofoil is another key factor in the design of actual aircraft aerofoil sections. For small aircraft this may be caused by propellor wake, and for all aircraft this can be caused by wind gusts as well as the wake of other planes. It is undesirable to have an aerofoil which will demonstrate a drastic change in characteristics under the influence of a wind gust or other unsteady feature. As such aerofoils that perform well without separation over a range of incidences and Reynolds number

will be more desirable than the absolute highest C_l/C_d possible but only for 1 very specific condition. As shown in 5.4 and 5.5, this can be achieved by increasing aerofoil thickness and by alleviating sharp peaks in the Cp distribution by smoothing the curvature distribution of the aerofoil.

4. The stall characteristics of a given section are also very important when considering the real world usability of a section. Despite the very high C_l/C_d values achieved by the 2 high Re aerofoils (Figure 12), they have a large suction peak shortly after the LE (Figure 13a). If a wind-gust were to increase the angle of attack momentarily, it is possible that this suction peak would become even larger and risk separation of the aerofoil shortly behind the suction peak due to the strong APG. Separation occurring close to the LE leads to a very dangerous phenomenon known as LE stall. It is particularly dangerous because, unlike TE stall where separation begins at the rear and slowly spreads forward, gradually reducing the section C_l, LE stall results in a catastrophic and very sudden reduction in the section C_l which can be very difficult to control and can lead to unrecoverable dives in some cases. It is therefore desirable to design an aerofoil without significant curvature and thus suction peaks close to the leading edge so that LE stall is discouraged and the safer stall mechanism of TE stall is more likely to occur first.

5. A final point of consideration in designing aerofoils for real world applications is the behaviour of the aerofoil over a range of flight conditions, most importantly the difference between the cruise condition at which we seek to optimise C_l/C_d, and at take-off and landing where we are not so much interested in section lift/drag ratio. There is a very large difference between the conditions at take-off and landing when compared to the cruise condition. Most noticeably, at higher altitudes, the density of air and is significantly lower than at ground level which would decrease the Re number at cruise. The competing factor is that at cruise the aircraft will be travelling much faster than at take-off or landing which would increase the Re number at cruise compared to take-off and landing. As such, we need our aerofoil to perform well over a wide range of Reynolds numbers, most noticeably we do not want the risk of dangerous LE stall at take-off and landing when the Reynolds number is different to the designed optimum cruise condition. This can be achieved by a smooth Cp distribution which is tolerant to a wider range of Reynolds numbers as discussed in 5.3 and 5.4.

6 Conclusions

1. The numerical model developed allows a significant number of aerofoil designs to be processed, validated, and compared in a much shorter period of time than more intensive methods such as 3D RANS CFD.

2. In fact, this type of numerical model is very similar to that employed in `JavaFoil` [4] which is used in industries for aerofoil design.

3. The accuracy of said model is sufficient to compare different section designs and also does show a difference in optimum designs for different Reynolds numbers. User care must be taken when employing the model to ensure that C_l/C_d values past the first instance of turbulent separation are not accurate as the calculated lift does not decrease upon turbulent separation as it actually would in a real viscous flow.

4. The design of aerofoil sections in 2D is clearly not suitable for final validation of whole wing designs and will not account for variations in aerofoil camber, thickness, and twist along the span of an aircraft. However, this report has shown it can be a very useful tool for primary analysis of a broad range of concepts.

5. As a general rule of thumb from the analysis, the absolute optimum C_l/C_d ratio aerofoil sections for high Reynolds numbers appear to have a larger peak in surface pressure gradient as the boundary layer is both higher energy and also often transitioned to turbulence before the suction peak which was not observed at low Reynolds numbers.

6. As a result of the above, aerofoils designed for low Reynolds number applications often performed well at high Reynolds number but the opposite was found not to be true in more circumstances.

7. Similarly, thinner aerofoils were found to exhibit a sharp but high peak in their C_l/C_d against α curves whereas thicker aerofoils exhibited a much broader peak - approaching a nearly flat line in the extreme.

8. This was found to be due to the larger-radius and smoother curvature around the leading edge which reduced any peaks in the surface Cp distribution over a much broader range of incidences and therefore delayed turbulent separation or lessened the severity at extreme incidences at the cost of absolute optimum C_l/C_d performance for a particular α.

9. Improvements to the model would likely focus on improving the accuracy of the lift solution as this is currently a pure inviscid solution and therefore ignores any separation completely which is clearly nonphysical as separation will change the curvature that a streamline experiences and therefore change the surface Cp distribution and calculated lift.

10. It may be possible to tackle this issue by evaluating the boundary layer thickness from the viscous boundary layer solver and adding this thickness normal to the underlying aerofoil surface to generate a new aerofoil surface. However, this would require an iterative solution process as first the boundary layer thickness must be calculated using a calculated inviscid Cp distribution. Then the boundary layer thickness can be added normal to the aerofoil surface and a new inviscid Cp distribution can be calculated. For improved accuracy, this may then need to be entered into the boundary layer solver again to calculate new separation locations and boundary layer thickness from which the inviscid Cp distribution can again be calculated. It is clear that this method would add significant complexity and may pose numerical instability issues also and was thus far outside the scope of this project.

11. A simpler but perhaps still valuable method to improve the lift solution accuracy is by applying a simple percentage reduction to the lift coefficient based on the percentage of the aerofoil chord that is experiencing turbulent separation. This would not be very accurate but would reduce the risk of accidentally designing an aerofoil section that appears to have an optimum C_l/C_d ratio but actually experiences significant separation and as such the lift is much lower than the model predicts.

12. Another very simple (and crude) method to avoid designing significantly separated sections is to simply stop calculating and plotting the C_l/C_d ratio if the turbulent separation location exceeds a certain fraction of the aerofoil chord.

13. A final improvement to the model could be made by including a compressibility correction to the Cp distribution (such as the Karman-Tsien correction [5]) for Mach numbers less than 0.5 (i.e where there is no risk of the flow exceeding Mach 1 over the suction surface of the aerofoil which would be the case for aerofoils in the transonic regime).

7 Bibliography

References

[1] *A computer program for the design and analysis of low-speed airfoils - NASA Technical Reports Server (NTRS) — ntrs.nasa.gov.* https://ntrs.nasa.gov/citations/19800020753. [Accessed 10-06-2024].

[2] Wei Wayen Chen, Kevin Chiu, and Mark Fuge. "Aerodynamic Design Optimization and Shape Exploration using Generative Adversarial Networks". In: Jan. 2019. DOI: 10.2514/6.2019-2351.

[3] Garcia-Mayoral. "Part IIA Project SA1—Aircraft Wing Analysis". In: (2024).

[4] *JavaFoil — mh-aerotools.de.* https://www.mh-aerotools.de/airfoils/javafoil.htm. [Accessed 10-06-2024].

[5] Roelof Vos and Saeed Farokhi. "Transonic Similarity Laws". In: *Introduction to Transonic Aerodynamics.* Dordrecht: Springer Netherlands, 2015, pp. 81–144.

[6] *WORTMANN FX 05-H-126 AIRFOIL (fx05h126-il) — airfoiltools.com.* http://airfoiltools.com/airfoil/details?airfoil=fx05h126-il. [Accessed 09-06-2024].

8 Appendices

8.1 Code Additions

Listing 1: curvature_vectors.m

```
function curvature_vectors = spline_curvature(xs, ys, toggle)
    %returns curvature vectors of cubic polynomial fit spline to the aerofoil
        surface spline
    %can then be plotted using matlab quiver function

    %Find piecewise polynomial spline curve
    splineCurve = spline(xs, ys);
    %evaluate first and second derivatives
    pp_der1 = fnder(splineCurve, 1);
    pp_der2 = fnder(splineCurve, 2);
    % Evaluate the first and second derivatives at the points given
    dy = ppval(pp_der1, xs);
    d2y = ppval(pp_der2, xs);

    % Compute the curvature
    if toggle == -1
        curvature = abs(d2y) ./ (1 + dy.^2).^(3/2);
    end
    if toggle == 1
        curvature = -d2y ./ (1 + dy.^2).^(3/2);
    end
    %by changing abs(d2y) to -d2y and removing the minus multiplication to all
        curvature_vectors_bot
    %the program can be modified to show reflexes in curvature either side of
        the surface
    %current setting is all on the outside of the surface

    % Compute the tangent vector to the aerofoil surface
    dx = ones(size(dy));
    tangent = [dx; dy];
    % Normalize the tangent vector
    tangent_norm = sqrt(tangent(1,:).^2 + tangent(2,:).^2);
    tangent_unit = tangent ./ tangent_norm;
    % Compute the normal vector by rotating the tangent vector by 90 degrees
    normal = [-tangent_unit(2,:); tangent_unit(1,:)];
    % Compute the curvature vectors
    curvature_vectors = curvature .* normal;
end
```

Listing 2: plot_upr_curve.m

```
function plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce)
    %plots upper surface curvature vectors for use in conjunction with
    %wasg_curvature_refined
    [~,LEpos] = mink(xs,1);
    xs_top = xs(:,1:curv_reduce:LEpos-1);
    ys_top = ys(:,1:curv_reduce:LEpos-1);
    curvature_vectors_top = spline_curvature(xs_top,ys_top,curv_tog);
    curvature_vectors_top = curvature_vectors_top/curv_scale;
    quiver(xs_top,ys_top, curvature_vectors_top(1,:), curvature_vectors_top(2,:)
        , 0,'-.r');
end
```

Listing 3: plot_lwr_curv.m

```
function plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce)
    %plots lower surface curvature vectors for use in conjunction with
    %wasg_curvature_refined
    [~,LEpos] = mink(xs,1);
    xs_bot = xs(:,LEpos+1:curv_reduce:end);
    ys_bot = ys(:,LEpos+1:curv_reduce:end);
    curvature_vectors_bot = spline_curvature(xs_bot,ys_bot,curv_tog);
    curvature_vectors_bot = curv_tog*curvature_vectors_bot/curv_scale;
    quiver(xs_bot,ys_bot, curvature_vectors_bot(1,:), curvature_vectors_bot(2,:)
        , 0,'-.r');
end
```

Listing 4: plot_ref_geom.m

```
function plot_ref_geom()
    %plots reference profile geometries for use in conjunction with
    %wasg_curvature_refined
    refs = {'naca4412', 'basg_3'}; %references to plot
    %refs = {'naca8406', 'naca8208', 'naca8412'}; %references to plot
    len_refs = length(refs);
    colours = {[0 0.5 0],"b","m","#4DBEEE","#77AC30","#7E2F8E","#EDB120","#0072
        BD","#D95319","#A2142F"};
    len_colours = length(colours);
    for i = 1:len_refs
        file = load(strcat('Geometry/', refs{i}, '.surf'));
        x=file(2:end-1,1);
        y=file(2:end-1,2);
        [xs, ys] = splinefit([1;x;1],[0;y;0],0);
        plot(xs,ys,"-", "Color", colours{mod(i,len_colours)}, "linewidth", 0.75)
    end
end
```

Listing 5: wasg_curvature_refined.m

```
function WASG
% CUED Wing Analysis Surface Generator
% FRJA2 - 2024 - addition of curvature and reference geometry plotting
% RGM - developed for SA1 2020
% based on 'ManipulateData.m' written by Lindo Ouseph.
%
% Manipulate an aerofoil section with the trailing edge fixed at
% (x,y)=(1,0) interactively. The aerofoil shape is constructed using
% splines based on a list of nodes.
%
% WASG is designed to be run on your main SA1 folder, one level above
% the 'Geometry' folder.
%
% List of functions:
%  left mouse click - select a node, or create a new one if clicking on
%                the aerofoil between points.
%  click and drag left mouse button - move the selected point.
%  arrow keys - move the selected point at a rate 'delta'.
%  right mouse click - delete the node clicked on.
%  u - undo last action (up to 20).
%  r - redo, cancel last undo.
%  z - zoom, create a secondary magnified figure, centred on the last
%      cursor position, that the control is
%      temporarily transferred to. Pressing z again closes the second
%      figure and returns control to the primary one. Functions 'l', 's',
%      'b', and 't' are not available while in zoom mode.
%  l - load aerofoil from a '.surf' file.
%  s - save aerofoil into a '.surf' file.
%  b - back up aerofoil layout into a memory buffer.
%  t - restore aerofoil from backup 'b'.
%  d - delta, reset the rate of displacement of nodes when using the arrow
%      keys.
%  v - show upper surface curvature vectors on aerofoil surface
%  x - show lower surface curvature
%  m - show reference geometries as specified in plot_ref_geom.m

clear
close all

b=0;p=0;
delta=1e-4; %change sensitivity of motion with arrow keys
max_undo = 20;

show_ref = false; %state flag for plotting reference geometry

%state flags for plotting upper and lower surface curvatures
upr_curv = false;
lwr_curv = false;

curv_tog = -1;
%-1 displays absolute curvature vectors always on the outside of the aerofoil
%1 displays curvature vectors on the outside of the aerofoil if concave and on
    the inside if convex

curv_scale = 100; %may need to change axes limits below
curv_reduce = 1; %reduce number of curvature vectors by this INTEGER factor


xmax = 1.2;
xmin = -0.5;
ymax =  .4;
ymin = -.2;


pathin=[pwd,'/Geometry/'];
[filein,pathin]=uigetfile([pathin '*.surf']);
y=load([pathin,filein]);
x=y(2:end-1,1);
y=y(2:end-1,2);
L=length(x);
I=(x-1).^2+y.^2; I=find(I==max(I)); I=I(1);
Xbk=x;
Ybk=y;hZ=[];
axisZ=[];
aZ=[];
deltaZ=.2*delta;
deltxt=8e-3;
deltxtZ=1e-3;
Xbk=x;Ybk=y;Lmax=L;
xundo=[];yundo=[];
Lundo=[];Iundo=[];
xredo=[];yredo=[];
Lredo=[];Iredo=[];
Res=get(0); Res=Res.ScreenSize;
h=figure('units','normalized',...
     'outerposition',[.05 .2 1.35*Res(4)/Res(3) .6],...
     'DockControls','off',...
     'MenuBar','none',...
     'name','Wing Analysis Section Generator (Matlab version)',...
     'NumberTitle','off',...
     'WindowKeyPressFcn',@Key,...
     'windowbuttondownfcn',@Down,...
     'WindowButtonMotionFcn',@Move,...
     'WindowButtonUpFcn',@Up,...
     'DeleteFcn',@figDelete);
a=axes('position',[.10,.10,.87,.87]);
[xs, ys] = splinefit([1;x;1],[0;y;0],0);

plot(xs,ys,'k', ...
        [1;x],[0;y],'.k', ...
        'markersize',13,'markerfacecolor','k');

hold on
if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
if show_ref; plot_ref_geom(); end
hold off


axis equal
axis([xmin xmax ymin ymax])
uicontrol('style','text','Fontsize',9, ...
     'position',[1 130 80 220],...
     'string',{'u-undo';'r-redo';'z-zoom';'l-load';'s-save'; ...
               'b-back up';'t-restore';'d-delta';'v-upr curv';...
               'x-lwr curv';'m-show ref'},...
     'foregroundcolor','k');


     %% @Down - what happens when a mouse button is pressed
     function Down(varargin)
             xundo=[[x;0*(length(x)+1:Lmax)'],xundo(:,1:min([size(xundo,2),
                 max_undo-1]))];
        yundo=[[y;0*(length(y)+1:Lmax)'],yundo(:,1:min([size(yundo,2),max_undo
             -1]))];
        Lundo=[L,Lundo(1:min([length(Lundo),max_undo-1]))];
        Iundo=[I,Iundo(1:min([length(Iundo),max_undo-1]))];
        p=get(a,'currentpoint');
        [V2, I]=min((x-p(1)).^2+(y-p(3)).^2);
        xx=x;
        yy=y;
        xx(I)=[];
        yy(I)=[];
        mindist2=min((xx-x(I)).^2+(yy-y(I)).^2);
        switch varargin{1}.SelectionType
        %% Mouse left-click:
        case 'normal'
             %% Add a knot if left-click is close to aerofoil contour:
             if V2>min(1e-4,mindist2)
                 [xs, ys] = splinefit([1;x;1],[0;y;0],1);
                 ppk = 100; % needs to be the same in 'splinefit.m'
                 [V2, Is]=min((xs-p(1)).^2+(ys-p(3)).^2);
                 if V2<1e-4
                     xI=xs(Is);yI=ys(Is);
                     I = ceil(Is/ppk);
                     L=L+1;
                     Lmax=max([L,Lmax]);
                     x(I+1:L)=x(I:end);
                     y(I+1:L)=y(I:end);
                     x(I)=xI;
                     y(I)=yI;
                     xundo=[xundo;0*(size(xundo,1)+1:Lmax)'*(1:size(xundo,2))];
                     yundo=[yundo;0*(size(yundo,1)+1:Lmax)'*(1:size(yundo,2))];
                     xredo=[];
                     yredo=[];
                     Lredo=[];
                     Iredo=[];
                     b=1;
                     [xs, ys] = splinefit([1;x;1],[0;y;0],0);
                     try delete(t);end

                     plot(xs,ys,'k', ...
                         [1;x],[0;y],'.k', ...
                             x(I),y(I),'xk','markersize',13)

                     hold on
                     if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,
                         curv_reduce); end
                     if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,
                         curv_reduce); end
                     if show_ref; plot_ref_geom(); end
                     hold off

                     axis equal
                     axis([xmin xmax ymin ymax])
                     t=text(x(I)+deltxt,y(I)-deltxt, ...
                             ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f')
                                 ,')'], ...
                             'color','k','fontweight','bold');
                     drawnow
                 end
             %% Select a knot if left-click is close to it:
             else
                 b=1;
                 %[xs ys] = splinefit([1;x;1],[0;y;0],0);%redundant?
                 try delete(t);end

                 plot(xs,ys,'k', ...
                         [1;x],[0;y],'.k', ...
                         x(I),y(I),'xk','markersize',13)

                 hold on
                 if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                     ); end
                 if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                     ); end
                 if show_ref; plot_ref_geom(); end
                 hold off

                 axis equal
                 axis([xmin xmax ymin ymax])
                 t=text(x(I)+deltxt,y(I)-deltxt, ...
                         ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f'),')'
                             ], ...
                         'color','k','fontweight','bold');
                 drawnow
             end
        %% Mouse right click:
        case 'alt'
```

```
        %%% Delete a knot if right-click is close to it:
        if V2<min([1e-4,mindist2])
            L=L-1;
            x=xx;
            y=yy;
            xredo=[];
            yredo=[];
            Lredo=[];
            Iredo=[];
            [xs, ys] = splinefit([1;x;1],[0;y;0],0);
            try delete(t);end

            plot(xs,ys,'k', ...
                [1;x],[0;y],'.k','markersize',13)

            hold on
            if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if show_ref; plot_ref_geom(); end
            hold off

            axis equal
            axis([xmin xmax ymin ymax])
            drawnow
        end
    end
end

%%% @Up - what happens when a mouse button is released
function Up(varargin)
    b=0;
end

%%% @Move - what happens when the mouse moves
%%% (while holding the left button clicked for b=true)
function Move(varargin)
    if b
        p=get(a,'currentpoint');
        x(I)=max([min([p(1),xmax]),xmin]);
        y(I)=max([min([p(3),ymax]),ymin]);
        xredo=[];
        yredo=[];
        [xs, ys] = splinefit([1;x;1],[0;y;0],0);
        try delete(t);end

        plot(xs,ys,'k', ...
                [1;x],[0;y],'.k', ...
                x(I),y(I),'xk','markersize',13)

        hold on
        if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce);
            end
        if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce);
            end
        if show_ref; plot_ref_geom(); end
        hold off

        axis equal
        axis([xmin xmax ymin ymax])
        t=text(x(I)+deltxt,y(I)-deltxt, ...
            ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f'),')'], ...
            'color','k','fontweight','bold');
        drawnow
    end
end

%% @Key - the various keyboard options
function Key(varargin)
    if ~delta;delta=mean(abs(diff(y)));end
    % to be done: if varargin{2}.Modifier=='control'
    switch varargin{2}.Key
        %% 'l' load a new .surf file
        case 'l'
            [filein,pathin]=uigetfile([pathin '*.surf']);
            y=load([pathin,filein]);
            x=y(2:end-1,1);
            y=y(2:end-1,2);
            L=length(x);
            I=(x-1).^2+y.^2; I=find(I==max(I)); I=I(1);
            Xbk=x;
            Ybk=y;
        %% 's' save into a .surf file:
        case 's'
            dataout=[[1;x;1],[0;y;0]];
            [fileout,pathout]=uiputfile([pathin '*.surf']);
            save([pathout fileout],'dataout','-ascii')
        %% 'b' backup configuration
        case 'b'
            Xbk=x;
            Ybk=y;
            disp('knot position backed up')
        %% 't' restore last backup
        case 't'
            x=Xbk;
            y=Ybk;
            L=length(x);
            I=1;
            disp('knot position restored')
        %% 'u' undo last action
        case 'u'
            if isempty(xundo)
                disp('undo max reached')
            else
                Lredo=[L,Lredo(1:min([length(Lredo),max_undo-1]))];
                Iredo=[I,Iredo(1:min([length(Iredo),max_undo-1]))];
                xredo=[[x;0*(length(x)+1:Lmax)'],xredo(:,1:min([size(xredo
                    ,2),max_undo-1]))];
                yredo=[[y;0*(length(y)+1:Lmax)'],yredo(:,1:min([size(yredo
                    ,2),max_undo-1]))];
                L=Lundo(1);
                I=Iundo(1);
                x=xundo(1:L,1);
                y=yundo(1:L,1);
                Lundo=Lundo(2:end);
                Iundo=Iundo(2:end);
                xundo=xundo(:,2:end);
                yundo=yundo(:,2:end);
            end
        %% 'r' redo - cancel last undo
        case 'r'
            if isempty(xredo)
                disp('last action reached')
            else
                Lundo=[L,Lundo(1:min([length(Lundo),max_undo-1]))];
                Iundo=[I,Iundo(1:min([length(Iundo),max_undo-1]))];
                xundo=[[x;0*(length(x)+1:Lmax)'],xundo(:,1:min([size(xundo
                    ,2),max_undo-1]))];
                yundo=[[y;0*(length(x)+1:Lmax)'],yundo(:,1:min([size(yundo
                    ,2),max_undo-1]))];
                L=Lredo(1);
                I=Iredo(1);
                x=xredo(1:L,1);
                y=yredo(1:L,1);
                Lredo=Lredo(2:end);
                Iredo=Iredo(2:end);
                xredo=xredo(:,2:end);
                yredo=yredo(:,2:end);
            end
        %% 'z' zoomed view
        case 'z'
            p=get(a,'currentpoint');
            axisZ=[p(1)-.05 p(1)+.05 p(3)-.03 p(3)+.03];
            try delete(t);end
            set(h,'WindowKeyPressFcn',@emptyFunctionHandle,...
                 'windowbuttondownfcn',@emptyFunctionHandle,...
                 'WindowButtonMotionFcn',@emptyFunctionHandle,...
                 'WindowButtonUpFcn',@emptyFunctionHandle)
            hZ=figure('units','normalized',...
                 'outerposition',[.3 .3 1.05*Res(4)/Res(3) .7],...
                 'DockControls','off',...
                 'MenuBar','none',...
                 'name','zoomed view',...
                 'NumberTitle','off',...
                 'WindowKeyPressFcn',@KeyZ,...
                 'windowbuttondownfcn',@DownZ,...
                 'WindowButtonMotionFcn',@MoveZ,...
                 'WindowButtonUpFcn',@UpZ,...
                 'DeleteFcn',@figDeleteZ);
            aZ=axes('position',[.10,.10,.87,.87]);
            plot(xs,ys,'k', ...
                    [1;x],[0;y],'.k', ...
                    x(I),y(I),'xk','markersize',13)
            axis equal
            axis(axisZ)
            drawnow
        %% press 'd' to change the rate of displacement of a knot
        %% when using the arrow keys instead of holding left mouse
        %% button
        case 'd'
            delta=str2double(inputdlg('enter delta'));
            deltaZ=.2*delta;
        case 'leftarrow'
            x(I)=max([x(I)-delta,xmin]);
        case 'rightarrow'
            x(I)=min([x(I)+delta,xmax]);
        case 'downarrow'
            y(I)=max([y(I)-delta,ymin]);
        case 'uparrow'
            y(I)=min([y(I)+delta,ymax]);
        case 'v'
            upr_curv = ~upr_curv;
            case 'x'
                lwr_curv = ~lwr_curv;
            case 'm'
                show_ref = ~show_ref;

        end
        [xs, ys] = splinefit([1;x;1],[0;y;0],0);
        set(0, 'CurrentFigure', h)
        try delete(t);end

        plot(xs,ys,'k', ...
                [1;x],[0;y],'.k', ...
                x(I),y(I),'xk','markersize',13)

        hold on
        if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
        if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
        if show_ref; plot_ref_geom(); end
        hold off

        axis equal
        axis([xmin xmax ymin ymax])
        t=text(x(I)+deltxt,y(I)-deltxt, ...
                ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f'),')'], ...
                'color','k','fontweight','bold');
        drawnow
end


%%%%%%%%%%%% FUNCTIONS FOR ZOOMED FIGURE %%%%%%%%%%%%
%%% @Down - what happens when a mouse button is pressed
function DownZ(varargin)
    xundo=[[x;0*(length(x)+1:Lmax)'],xundo(:,1:min([size(xundo,2),max_undo
        -1]))];
    yundo=[[y;0*(length(y)+1:Lmax)'],yundo(:,1:min([size(yundo,2),max_undo
        -1]))];
    Lundo=[L,Lundo(1:min([length(Lundo),max_undo-1]))];
    Iundo=[I,Iundo(1:min([length(Iundo),max_undo-1]))];
    p=get(aZ,'currentpoint');
    [V2, I]=min((x-p(1)).^2+(y-p(3)).^2);
    xx=x;
    yy=y;
    xx(I)=[];
    yy(I)=[];
    mindist2=min((xx-x(I)).^2+(yy-y(I)).^2);
    switch varargin{1}.SelectionType
    %%% Mouse left-click:
    case 'normal'
        %%% Add a knot if left-click is close to aerofoil contour:
        if V2>min(1e-6,mindist2)
            [xs, ys] = splinefit([1;x;1],[0;y;0],1);
            ppk = 100; % needs to be the same in 'splinefit.m'
            [V2, Is]=min((xs-p(1)).^2+(ys-p(3)).^2);
            if V2<1e-6
                xI=xs(Is);yI=ys(Is);
                I = ceil(Is/ppk);
                L=L+1;
                Lmax=max([L,Lmax]);
                x(I+1:L)=x(I:end);
                y(I+1:L)=y(I:end);
                x(I)=xI;
                y(I)=yI;
                xundo=[xundo;0*(size(xundo,1)+1:Lmax)'*(1:size(xundo,2))];
                yundo=[yundo;0*(size(yundo,1)+1:Lmax)'*(1:size(yundo,2))];
                xredo=[];
                yredo=[];
                b=1;
                [xs, ys] = splinefit([1;x;1],[0;y;0],0);%redundant?
                set(0, 'CurrentFigure', h)

                plot(xs,ys,'k', ...
                        [1;x],[0;y],'.k', ...
                        x(I),y(I),'xk','markersize',13);

                hold on
                if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,
                    curv_reduce); end
                if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,
                    curv_reduce); end
                if show_ref; plot_ref_geom(); end
                hold off

                axis equal
                axis([xmin xmax ymin ymax])
                %drawnow
                figure(hZ)
                try delete(t);end

                plot(xs,ys,'k', ...
                        [1;x],[0;y],'.k', ...
                        x(I),y(I),'xk', ...
                        'markersize',13);

                hold on
                if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,
                    curv_reduce); end
                if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,
                    curv_reduce); end
                if show_ref; plot_ref_geom(); end
                hold off

                axis equal
                axis(axisZ)
                t=text(x(I)+deltxtZ,y(I)-deltxtZ, ...
                        ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f')
                            ,')'], ...
                        'color','k','fontweight','bold');
                drawnow
            end
        %%% Select a knot if left-click is close to it:
        else
            b=1;
            [xs, ys] = splinefit([1;x;1],[0;y;0],0);%redundant?
            set(0, 'CurrentFigure', h)

            plot(xs,ys,'k', ...
                [1;x],[0;y],'.k', ...
                x(I),y(I),'xk', ...
                'markersize',13);

            hold on
            if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if show_ref; plot_ref_geom(); end
            hold off

            axis equal
            axis([xmin xmax ymin ymax])
            %drawnow
            figure(hZ)
            try delete(t);end

            plot(xs,ys,'k', ...
                [1;x],[0;y],'.k', ...
                x(I),y(I),'xk', ...
                'markersize',13);

            hold on
            if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if show_ref; plot_ref_geom(); end
            hold off

            axis equal
            axis(axisZ)
            t=text(x(I)+deltxtZ,y(I)-deltxtZ, ...
                    ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f'),')'
                        ], ...
                    'color','k','fontweight','bold');
            drawnow
        end
    %%% Mouse right click:
    case 'alt'
        %%% Delete a knot if right-click is close to it:
        if V2<min([1e-6,mindist2])
            L=L-1;
            x=xx;
            y=yy;
            xredo=[];
            yredo=[];
            [xs, ys] = splinefit([1;x;1],[0;y;0],0);
            set(0, 'CurrentFigure', h)

            plot(xs,ys,'k', ...
                    [1;x],[0;y],'.k', ...
                    'markersize',13);

            hold on
            if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if show_ref; plot_ref_geom(); end
            hold off

            axis equal
            axis([xmin xmax ymin ymax])
            %drawnow
            figure(hZ)
```

```
            try delete(t);end

            plot(xs,ys,'k', ...
                    [1;x],[0;y],'.k', ...
                    'markersize',13);

            hold on
            if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce
                ); end
            if show_ref; plot_ref_geom(); end
            hold off

            axis equal
            axis(axisZ)
            drawnow
        end
    end
end

%%% @Up - what happens when a mouse button is released
function UpZ(varargin)
    b=0;
end

%%% @Move - what happens when the mouse moves
%%% (while holding the left button clicked for b=true)
function MoveZ(varargin)
    p=get(aZ,'currentpoint');
    if b
        x(I)=max([min([p(1),xmax]),xmin]);
        y(I)=max([min([p(3),ymax]),ymin]);
        xredo=[];
        yredo=[];
        [xs, ys] = splinefit([1;x;1],[0;y;0],1);
        set(0, 'CurrentFigure', h)

        plot(xs,ys,'k', ...
                [1;x],[0;y],'.k', ...
                x(I),y(I),'xk','markersize',13);

        hold on
        if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce);
            end
        if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce);
            end
        if show_ref; plot_ref_geom(); end
        hold off

        axis equal
        axis([xmin xmax ymin ymax])
        %drawnow
        set(0, 'CurrentFigure', hZ)
        try delete(t);end

        plot(xs,ys,'k', ...
                [1;x],[0;y],'.k', ...
                x(I),y(I),'xk', ...
                'markersize',13);
        hold on
        if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce);
            end
        if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce);
            end
        if show_ref; plot_ref_geom(); end
        hold off

        axis equal
        axis(axisZ)
        t=text(x(I)+deltxtZ,y(I)-deltxtZ, ...
                   ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f'),')'
                       ], ...
                   'color','k','fontweight','bold');
        drawnow
    end
end

%% @Key - the various keyboard options
function KeyZ(varargin)
    if ~delta;delta=mean(abs(diff(y)));end
    % to be done: if varargin{2}.Modifier=='control'
    switch varargin{2}.Key
        %% 'u' undo last action
        case 'u'
            if isempty(xundo)
                disp('undo max reached')
            else
                Lredo=[L,Lredo(1:min([length(Lredo),max_undo-1]))];
                Iredo=[I,Iredo(1:min([length(Iredo),max_undo-1]))];
                xredo=[[x;0*(length(x)+1:Lmax)'],xredo(:,1:min([size(xredo
                    ,2),max_undo-1]))];
                yredo=[[y;0*(length(y)+1:Lmax)'],yredo(:,1:min([size(yredo
                    ,2),max_undo-1]))];
                L=Lundo(1);
                I=Iundo(min([length(Iundo),2]));
                x=xundo(1:L,1);
                y=yundo(1:L,1);
                Lundo=Lundo(2:end);
                Iundo=Iundo(2:end);
                xundo=xundo(:,2:end);
                yundo=yundo(:,2:end);
            end
        %% 'r' redo - cancel last undo
        case 'r'
            if isempty(xredo)
                disp('last action reached')
            else
                Lundo=[L,Lundo(1:min([length(Lundo),max_undo-1]))];
                Iundo=[I,Iundo(1:min([length(Iundo),max_undo-1]))];
                xundo=[[x;0*(length(x)+1:Lmax)'],xundo(:,1:min([size(xundo
                    ,2),max_undo-1]))];
                yundo=[[y;0*(length(x)+1:Lmax)'],yundo(:,1:min([size(yundo
                    ,2),max_undo-1]))];
                L=Lredo(1);
                I=Iredo(1);
                x=xredo(1:L,1);
                y=yredo(1:L,1);
                Lredo=Lredo(2:end);
                Iredo=Iredo(2:end);
                xredo=xredo(:,2:end);
                yredo=yredo(:,2:end);
            end
        case 'z'
            close(hZ)
            set(h,'WindowKeyPressFcn',@Key,...
                    'windowbuttondownfcn',@Down,...
                    'WindowButtonMotionFcn',@Move,...
                    'WindowButtonUpFcn',@Up)
            return
        %%% press 'd' to change the rate of displacement of a knot
        %%% when using the arrow keys instead of holding left mouse
        %%% button
        case 'd'
            delta=str2double(inputdlg('enter delta'));
            deltaZ=.2*delta;
        case 'leftarrow'
            x(I)=max([x(I)-delta,xmin]);
        case 'rightarrow'
            x(I)=min([x(I)+delta,xmax]);
        case 'downarrow'
            y(I)=max([y(I)-delta,ymin]);
        case 'uparrow'
            y(I)=min([y(I)+delta,ymax]);
        case 'v'
            upr_curv = ~upr_curv;
        case 'x'
            lwr_curv = ~lwr_curv;
        case 'm'
            show_ref = ~show_ref;
    end
    if I<1
        I=1;
    elseif I>L
        I=L;
    end
    [xs, ys] = splinefit([1;x;1],[0;y;0],1);
    set(0, 'CurrentFigure', h)

    plot(xs,ys,'k', ...
            [1;x],[0;y],'.k', ...
            x(I),y(I),'xk', ...
            'markersize',13);

    hold on
    if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
    if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
    if show_ref; plot_ref_geom(); end
    hold off

    axis equal
    axis([xmin xmax ymin ymax])
    %drawnow
    set(0, 'CurrentFigure', hZ)
    try delete(t);end

    plot(xs,ys,'k', ...
            [1;x],[0;y],'.k', ...
            x(I),y(I),'xk', ...
            'markersize',13);

    hold on
```

```
        if upr_curv; plot_upr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
        if lwr_curv; plot_lwr_curv(xs,ys,curv_tog,curv_scale,curv_reduce); end
        if show_ref; plot_ref_geom(); end
        hold off

        axis equal
        axis(axisZ)
        t=text(x(I)+deltxtZ,y(I)-deltxtZ, ...
                    ['(',num2str(x(I),'%8.4f'),',',num2str(y(I),'%8.4f'),')'],
                        ...
                    'color','k','fontweight','bold');
        drawnow
    end

    function figDelete(varargin)
        try close(hZ);end
    end

    function figDeleteZ(varargin)
        set(h,'WindowKeyPressFcn',@Key,...
                'Windowbuttondownfcn',@Down,...
                'WindowButtonMotionFcn',@Move,...
                'WindowButtonUpFcn',@Up)
        return
    end

    %%% empty function to make figure 'h' inactive while 'hZ' is active
    function emptyFunctionHandle(varargin)
    end

end
```