Aerodynamics Homework 07

Zakary Steenhoek

December 05, 2024

I. Part 1

For this part, the data is generated using the standard NASA computational fluid dynamics code that was used in previous assignments.

1. Question 1:

Using the provided NACA 0012 data files for friction and pressure coefficient found on Canvas:

a) Plot C_f vs x for angle of attack equal to 0 degrees and 15 degrees.

b) Using MATLAB, estimate the lift and drag coefficients per unit span on this airfoil for $\alpha = 0°$ and $\alpha = 15°$.

Assume: Compressible, viscous flow (RANS data). Thin airfoil. Dimensionless data.
Given: TACAA RANS pressure and friction data files for NACA 0012 at an angle of attack of 0 and 15 degrees.
Solution:

First, the Canvas data files are downloaded and stored on the MATLAB path, which is variable defined for reference. Some file names are manipulated for ease of parsing. The directory of NACA 0012 files is loaded for manipulation, and data structures to contain the extracted data and calculated data are initialized. A loop iterates over each file name in the file list, and determines if it is a file of interest, i.e. a friction or pressure coefficient data file at an angle of attack of 0 degrees or 15 degrees.

Each file of interest is categorized, the file data is loaded, and the filename is split to extract the angle of attack, which is then converted to an integer, and finally to radians. The columns needed for x-coordinate, z-coordinate, and either the friction or pressure coefficient are extracted, ensuring to use the command ‘flipud()’ to maintain standard practice for tabulating airfoil data. The result of this process is an organized data structure containing all the needed coordinates and coefficients for the NACA 0012 at an angle of attack of 0 degrees and 15 degrees.

The next step is to process the data to extract the force coefficients. The distinction needs to be made between the lift and drag coefficients from both the pressure and friction data for later analysis, and as such, it is easier to perform separate initial force calculations. The following equations are used to find the x and z body force coefficients:

$$C_z = \int_{airfoil} -C_p \ dx + \int_{airfoil} C_f \ dz$$

$$C_\mathrm{x} = \int_{airfoil} C_p \ d\mathrm{z} + \int_{airfoil} C_f \ d\mathrm{x}$$

Each of these integrals was calculated individually using the MATLAB built-in function 'trapz()', and these parts were summed to obtain the total force coefficients. These values are stored so they may be used to find the lift and drag coefficients, which is the simple process of converting from body axes to wind axes using the angle of attack. The following equations were used to convert to wind axes and determine the lift and drag coefficients:

$$C_l = C_z \cdot \cos\alpha - C_x \cdot \sin\alpha$$

$$C_\mathrm{d} = C_z \cdot \sin\alpha + C_x \cdot \cos\alpha$$

These equations were used on each of the individual force calculation parts to determine the lift and drag coefficients solely due to pressure and friction, and then used once more to determine the total lift and drag coefficient per unit span for the NACA 0012 at an angle of attack of 0 degrees and 15 degrees.

Finally, plots were generated for friction coefficient vs. dimensionless chord for each angle of attack, ensuring to differentiate between the upper and lower surfaces, as denoted in a legend.

Results:

The friction coefficient vs. dimensionless chord plot for NACA 0012 at an angle of attack equal to 0 degrees is shown below:

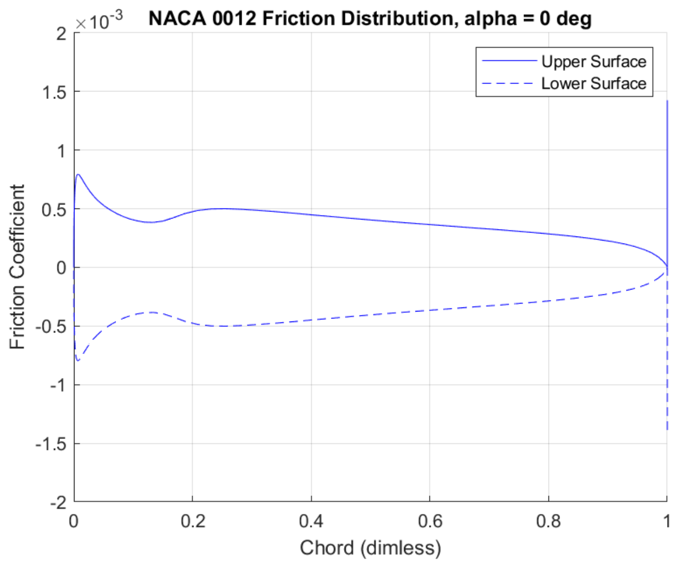

Figure [1.1.1]: friction coefficient vs. dimensionless chord for NACA 0012 at an angle of attack equal to 0 degrees

The friction coefficient vs. dimensionless chord plot for NACA 0012 at an angle of attack equal to 15 degrees is shown below:

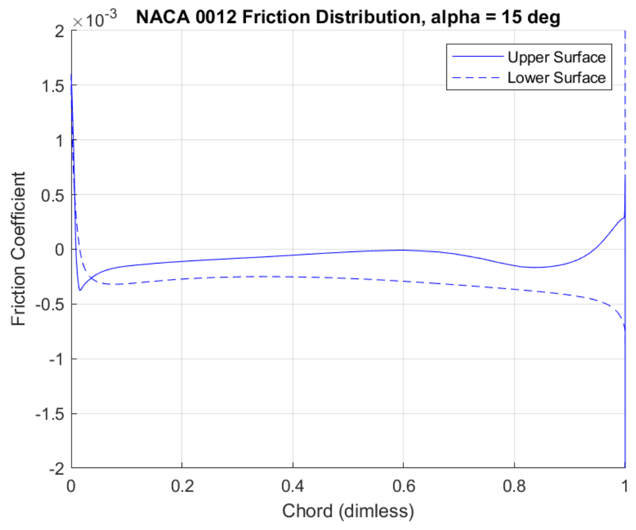

Figure [1.1.2]: friction coefficient vs. dimensionless chord for NACA 0012 at an angle of attack equal to 15 degrees

Next, the lift and drag coefficients due to pressure, friction, and the total lift and drag coefficients are tabulated for NACA 0012 at an angle of attack equal to 0 degrees:

C_l and C_d, NACA 0012, α = 0°		
Component	C_l	C_d
Friction	$-2.61085 \cdot \mathrm{E}^{-8}$	0.000763195
Pressure	−0.00156005	0.00241696
Total	−0.00156007	0.00318015

Table [1.1.1]: Lift and drag coefficients because of friction, pressure, and total NACA 0012 lift and drag coefficients at an angle of attack equal to 0 degrees

Next, the lift and drag coefficients due to pressure, friction, and the total lift and drag coefficients are tabulated for NACA 0012 at an angle of attack equal to 15 degrees:

C_l and C_d, NACA 0012, α = 15°		
Component	C_l	C_d
Friction	$-2.22088 \cdot E^{-5}$	0.000236049
Pressure	0.515149	0.150195
Total	0.515127	0.150431

Table [1.1.2]: Lift and drag coefficients because of friction, pressure, and total NACA 0012 lift and drag coefficients at an angle of attack equal to 15 degrees

Discussion:

For the NACA 0012 at a 0-degree angle of attack, it is well known that it will not generate lift, since there is no pressure differential. Thus, it makes sense from this standpoint that the lift coefficient contribution from the drag due to friction is essentially zero, as any force along the z dimension will be cancelled due to symmetry.

It is known from the Falkner-Skan equation that the shear stress on the surface is a function of the pressure gradient, which is a function of the curvature of the surface. When the surface curves away from the flow, the pressure gradient is negative, the boundary layer is slowing, and shear is reduced. The opposite is true for when the surface curves towards the flow. Consider the known equation for the friction coefficient:

$$C_f = \frac{\tau_\omega}{\frac{1}{2}\rho V_\infty^2}$$

However, the friction coefficient is also inversely related to the square of the flow speed. Thus, the friction coefficient is a representation of the relationship of surface curvature and flow speed. At the leading edge, near the stagnation point, the velocity is small, and even though the surface is very curved, the low flow speed dominates and results in a large friction coefficient. The flow speeds up rapidly as it rounds the leading edge, and a drop in friction coefficient is seen due to the increase in velocity. At roughly 0.12 chord, the friction coefficient becomes minimum, due to the high velocity and thin boundary layer. As the flow progresses over the rest of the airfoil, the negative curvature and thickening of the boundary layer reduces the shear, which is the dominating factor through this range. Due to symmetry, the case is the same on the lower surface as well. It is important to note here that the friction coefficient is not actually negative along the lower surface, as this would mean it is generating thrust. But it is a good way to represent the *magnitude* of the friction coefficient over the surfaces, which is the important factor.

The magnitude of the friction coefficient along the surface is what determines the total drag coefficient. Like how the area between the curve of the pressure coefficient is what determines the pressure force in the x and z dimensions due to drag, the area between the curves of the friction coefficient across the surfaces determines the friction force in the x and z directions due to pressure.

The friction force in the z direction is equal and opposite, but the friction force in the x-direction is not, rather, is a resultant of both the upper and lower surfaces. Looking at Fig. [1.1.2], the same principles mentioned above that govern the shape of the drag coefficient still apply, however the area between the upper and lower surface components is significantly smaller. This is why there is a smaller contribution to overall drag due to friction for the NACA 0012 at a 15 degree angle of attack. The *magnitude* of the friction across *both* surfaces is smaller, resulting in less overall drag contribution, which is evident when comparing Tab. [1.1.1] and Tab. [1.1.2]. The reason behind the smaller magnitude is once again a function of flow velocity and the relative slope of the surface of the airfoil. The increase in angle of attack provides more lift obviously, however the lack of symmetry that produces this drag also means that there is a much greater contribution to lifting force due to drag. Note that 'much greater' is incredibly relative, as at both angles of attack the overall contribution is still negligible, but the relative increase from a 0-degree AoA to a 15-

degree AoA is on the order of 10^3. Also, note that the sharp increase at the trailing edge is due to the lack of enforcement of the Kutta condition in data processing, and the friction at this point is only this insanely high in theory.

The pressure drag contribution has been analyzed prior, but the same results are shown here, where there is a very minor amount at a zero-degree angle of attack, once again due to symmetry. This contribution is much greater when the angle of attack is increased, and due to the pressure differential that generates lift, and the conversion from body to wind axes, the airfoil experiences much more pressure drag. The overall contributions show that, for this case, the pressure drag remains the main contributing factor, to total drag, *especially* when the airfoil is generating lift.

2. Question 2:

Using the provided NACA 4412 data files for friction and pressure coefficient found on Canvas:

a) Plot C_f vs x for angle of attack equal to 0 degrees and 15 degrees.

b) Using MATLAB, estimate the lift and drag coefficients per unit span on this airfoil for $\alpha = 0°$ and $\alpha = 15°$.

Assume: Compressible, viscous flow (RANS data). Thin airfoil. Dimensionless data.
Given: TACAA RANS pressure and friction data files for NACA 4412 at an angle of attack of 0 and 15 degrees.
Solution:

First, the Canvas data files are downloaded and stored on the MATLAB path, which is variable defined for reference. Some file names are manipulated for ease of parsing. The directory of NACA 4412 files is loaded for manipulation, and data structures to contain the extracted data and calculated data are initialized. A loop iterates over each file name in the file list, and determines if it is a file of interest, i.e. a friction or pressure coefficient data file at an angle of attack of 0 degrees or 15 degrees.

Each file of interest is categorized, the file data is loaded, and the filename is split to extract the angle of attack, which is then converted to an integer, and finally to radians. The columns needed for x-coordinate, z-coordinate, and either the friction or pressure coefficient are extracted, ensuring to use the command 'flipud()' to maintain standard practice for tabulating airfoil data. The result of this process is an organized data structure containing all the needed coordinates and coefficients for the NACA 4412 at an angle of attack of 0 degrees and 15 degrees.

The next step is to process the data to extract the force coefficients. The distinction needs to be made between the lift and drag coefficients from both the pressure and friction data for later analysis, and as such, it is easier to perform separate initial force calculations. The following equations are used to find the x and z body force coefficients:

$$C_z = \int_{airfoil} -C_p \ dx + \int_{airfoil} C_f \ dz$$

$$C_\mathrm{x} = \int_{airfoil} C_p \ \mathrm{d}z + \int_{airfoil} C_f \ d\mathrm{x}$$

Each of these integrals was calculated individually using the MATLAB built-in function 'trapz()', and these parts were summed to obtain the total force coefficients. These values are stored so they may be used to find the lift and drag coefficients, which is the simple process of converting from body axes to wind axes using the angle of attack. The following equations were used to convert to wind axes and determine the lift and drag coefficients:

$$C_l = C_z \cdot \cos\alpha - C_x \cdot \sin\alpha$$

$$C_\mathrm{d} = C_z \cdot \sin\alpha + C_x \cdot \cos\alpha$$

These equations were used on each of the individual force calculation parts to determine the lift and drag coefficients solely due to pressure and friction, and then used once more to determine the total lift and drag coefficient per unit span for the NACA 4412 at an angle of attack of 0 degrees and 15 degrees.

Finally, plots were generated for friction coefficient vs. dimensionless chord for each angle of attack, ensuring to differentiate between the upper and lower surfaces, as denoted in a legend.

Results:

The friction coefficient vs. dimensionless chord plot for NACA 4412 at an angle of attack equal to 0 degrees is shown below:

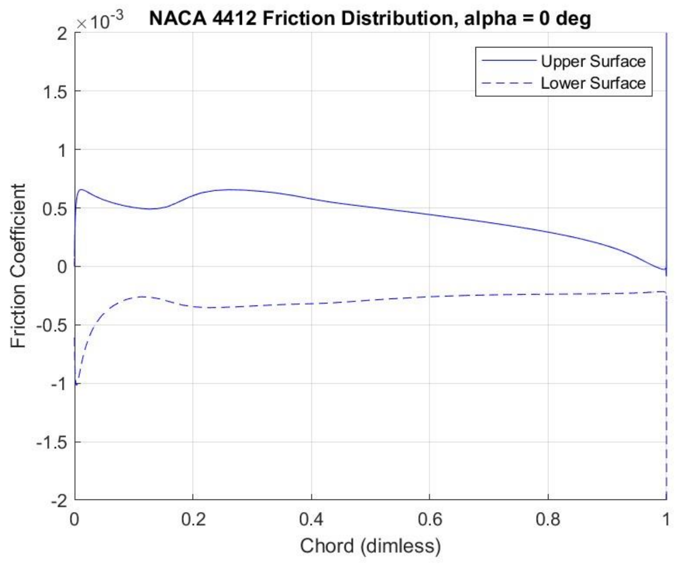

Figure [1.2.1]: friction coefficient vs. dimensionless chord for NACA 4412 at an angle of attack equal to 0 degrees

The friction coefficient vs. dimensionless chord plot for NACA 4412 at an angle of attack equal to 15 degrees is shown below:

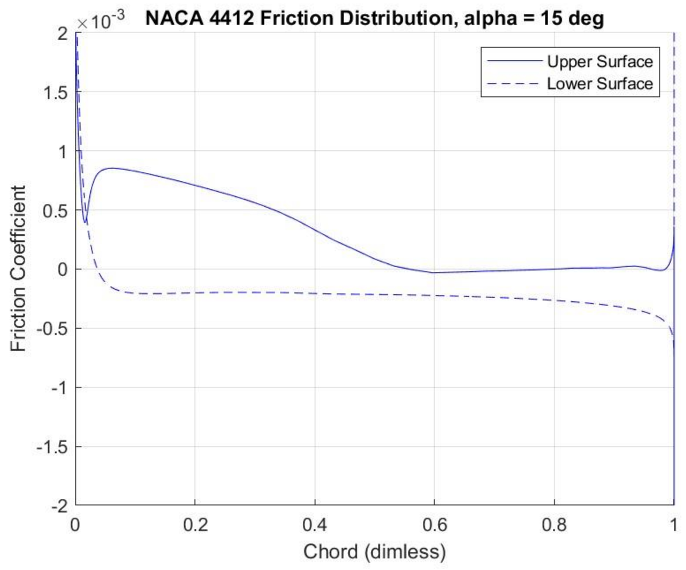

Figure [1.2.2]: friction coefficient vs. dimensionless chord for NACA 4412 at an angle of attack equal to 15 degrees

Next, the lift and drag coefficients due to pressure, friction, and the total lift and drag coefficients are tabulated for NACA 4412 at an angle of attack equal to 0 degrees:

C_l and C_d, NACA 4412, $\alpha = 0°$		
Component	C_l	C_d
Friction	$1.66260 \cdot \mathrm{E}^{-5}$	0.000741663
Pressure	0.380120	0.00322594
Total	0.380137	0.00396760

Table [1.2.1]: Lift and drag coefficients because of friction, pressure, and total NACA 4412 lift and drag coefficients at an angle of attack equal to 0 degrees

Next, the lift and drag coefficients due to pressure, friction, and the total lift and drag coefficients are tabulated for NACA 4412 at an angle of attack equal to 15 degrees:

C_l and C_d, NACA 4412, $\alpha = 15°$		
Component	C_l	C_d
Friction	$-2.85469 \cdot E^{-6}$	0.000507992
Pressure	1.59591	0.0348641
Total	1.59591	0.0353721

Table [1.2.2]: Lift and drag coefficients because of friction, pressure, and total NACA 4412 lift and drag coefficients at an angle of attack equal to 15 degrees

Discussion:

Note that the discussion in this section is majorly like the one prior, but some changes and additional thoughts are included due to camber and different zero-lift angle.

For the NACA 4412 at a 0-degree angle of attack, it is well known that it *will* generate lift, since there is a pressure differential. Thus, it makes sense that the lift contribution is not nearly as small as it was for the NACA 0012 at this AoA, but also not nearly as large as it was for the NACA 0012 at a 15-degree AoA.

This airfoil presents some very similar graphical results to the NACA 0012. At the leading edge, near the stagnation point, the velocity is small, and even though the surface is very curved, the low flow speed dominates and results in a large friction coefficient. The flow speeds up rapidly as it rounds the leading edge, and a drop in friction coefficient is seen due to the increase in velocity. At roughly 0.12 chord, the friction coefficient becomes minimum, due to the high velocity and thin boundary layer. As the flow progresses over the rest of the airfoil, the negative curvature and thickening of the boundary layer reduces the shear, which is the dominating factor through this range. This airfoil is not symmetric, so neither is the graph, but the difference is not super large, and the same general shape is seen. It is important to note here that the friction coefficient is still not actually negative along the lower surface, as this would mean it is generating thrust. But it is a good way to represent the *magnitude* of the friction coefficient over the surfaces, which is the important factor.

The magnitude of the friction coefficient along the surface is what determines the total drag coefficient. Like how the area between the curve of the pressure coefficient is what determines the pressure force in the x and z dimensions due to drag, the area between the curves of the friction coefficient across the surfaces determines the friction force in the x and z directions due to pressure.

Because the airfoil is generating lift in both cases, a dissymmetry exists for both cases. There is also a larger overall magnitude of surface curvature due to camber, and thus the relationship between flow speed and surface curvature is more complex, and the results show that the area between the curves is greater overall. At a 0-degree AoA, the difference in friction drag contribution is not super great, but the difference at a 15-degree AoA is notable. The NACA 4412 experiences seemingly a fair amount more drag here. Note that the force in the z direction seems to experience the opposite trend that was seen in the NACA 0012, however the magnitude is much smaller here. Also, note that the sharp increase at the trailing edge is due to the lack of enforcement of the Kutta condition in data processing, and the friction at this point is only this insanely high in theory. Still, the overall drag contribution comes from the pressure drag, even in the 0-degree AoA case.

II. Part 2

For this part of the homework assignment, you should use the MATLAB function falkner_skan.m. Be sure to explain how you solved each part.

1. Question 1:

For the case of a laminar boundary layer on a flat plate:

a) *Find an expression for the boundary layer thickness (δ) as a function of x and Re_x. The accepted expression is:*

$$\delta = \frac{5x}{\sqrt{Re_x}}$$

Comment on the consistency of your result vs. this value.

b) *Find an expression for the friction coefficient (C_f) as a function of Re_x.*

c) *Find the average skin friction coefficient, $\overline{C_f}$, as a function of Reynolds number based on chord length. Note that the accepted value for this is given as:*

$$\overline{C_f} = \frac{1.328}{\sqrt{Re_x}}$$

d) *Find an expression for the displacement thickness of a flat-plate boundary layer in terms of Re_x.*

Assume: Incompressible, Newtonian fluid. Steady, two-dimensional flow. Constant viscosity. Thin boundary layer, no flow separation. No slip condition, no flow penetration at the wall, boundary layer velocity approaches free-stream velocity at outer edge. No pressure gradient.

Given: MATLAB script 'falkner_skan.m', accepted expressions for δ and $\overline{C_f}$, as stated above. $\beta = 0$.

Note: A minor error in notation, where the boundary layer velocity at the edge of the boundary layer should be represented as u_e, is represented as u_∞, not to be confused with the freestream velocity U_∞

Solution:

For the case of a laminar boundary layer over a flat plate, the streamwise boundary layer velocity profile can be determined by solving the Falkner-Skan equation for $\beta = 0$, which is analogous to the classical Blasius equation solution. This involves the simultaneous solution of the continuity equation and the x-momentum differential equations. The energy equation can be omitted from this solution under the assumption that the flow is steady and incompressible, i.e. low speed flow conditions. The continuity [2.1] and momentum [2.2] equations are seen below, in their simplified forms after applying the assumptions listed above:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.1}$$

$$\rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} = \rho_\infty u_\infty \frac{du_\infty}{dx} + \mu \frac{\partial^2 u}{\partial y^2} \tag{2.2}$$

To obtain a solution to the system of differential equations, boundary conditions shall be set. In this case, the flow behavior at the wall, $y = 0$, and the flow behavior at the edge of the boundary layer can be examined. At the wall, the no-slip condition must be satiated, i.e. there is zero streamwise velocity at $y = 0$, and flow cannot penetrate the wall, i.e. there is zero transverse velocity at the wall. At a distance far away from the wall, the streamwise velocity equals the freestream velocity. These are shown below:

$$u(x, 0) = 0 \quad [2.3.1]$$

$$v(x, 0) = 0 \quad [2.3.2]$$

$$u(x, \infty) = u_\infty(x) \quad [2.3.3]$$

Knowing these boundary conditions, dimensionless parameters are defined, and coordinate transformations are used to simplify the solution. These encapsulate the multiple dimensional variables, including the x-coordinate, y-coordinate, constant kinematic viscosity, freestream velocity, and the variation of streamwise velocity through the boundary layer. The velocity profile is expressed as a fractional value between 0 and 1. This can be done because our boundary conditions for streamwise flow state that u is 0 at the surface, and $1 \cdot u_\infty$ at the edge of the boundary layer. The dimensional y-coordinate is represented as the dimensionless coordinate η, which is proportional to $\frac{y}{\delta}$. These are seen below:

$$\frac{u}{u_\infty}$$

$$\eta = y\sqrt{\frac{u_\infty}{2vx}} \quad [2.4.1]$$

Let the variable s be the coordinate transform mentioned above, which accounts for the effects of any variation in u_∞ on streamwise growth of the boundary layer, and is equal to the following:

$$s = \int u_\infty \; dx$$

This allows η to be expressed as:

$$\eta = \frac{u_\infty y}{\sqrt{2vs}} \quad [2.4.2]$$

The next step is to define a stream function that represents the boundary layer flow. This is allowable since we are assuming two-dimensional flow of an incompressible fluid. Stream representation allows both velocity component variables to be represented using a single function, further simplifying the solution. Note that the velocity potential representation cannot be used here, as the existence of the boundary layer we are analyzing implies non-negligible viscosity, and thus irrotationality cannot be proven. Each velocity component transformation is seen below:

$$u = \left(\frac{\partial \psi}{\partial y}\right) \quad [2.5.1]$$

$$v = -\left(\frac{\partial \psi}{\partial x}\right) \quad [2.5.2]$$

The stream function representation automatically satisfies the continuity condition, Eq. [2.1]. This leaves only the momentum equation [2.2] to solve, in terms of a single unknown, the stream function, which was defined above. The next step is to transform these from x and y coordinates to s and η coordinates.

This is done by firstly transforming partial derivative terms w.r.t. x and y, which can then be applied to Eq. [2.2] and Eq. [2.5]. This is done using Eq. [2.4] and applying the chain rule, as seen below:

$$\frac{\partial}{\partial y} = \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} = \frac{u_\infty}{\sqrt{2\nu s}}\frac{\partial}{\partial \eta} \qquad [2.6.1]$$

$$\frac{\partial}{\partial x} = \frac{\partial s}{\partial x}\left[\frac{\partial \eta}{\partial s}\frac{\partial}{\partial \eta} + \frac{\partial}{\partial s}\right] \qquad [2.6.2]$$

The streamwise velocity component, Eq. [2.5.1], can therefore be represented as seen below, using Eq. [2.6.1]:

$$u = \frac{u_\infty}{\sqrt{2\nu s}}\frac{\partial \psi}{\partial \eta}$$

Before converting the transverse velocity component, an expression for a transformed stream function is introduced as f, and is defined such that its derivative represents the dimensionless velocity profile only as a function of η:

$$u = u_\infty \frac{\partial f}{\partial \eta} \qquad [2.7.1]$$

Thus, the stream function can be represented as:

$$f = \frac{1}{\sqrt{2\nu s}}\psi$$

The transverse velocity component can now be transformed using the above derivations, as seen below:

$$v = -\frac{\partial s}{\partial x}\left[\frac{\partial \eta}{\partial s}\frac{\partial \psi}{\partial \eta} + \frac{\partial \psi}{\partial s}\right] = -u_\infty\sqrt{2\nu s}\left[\frac{\partial \eta}{\partial s}\frac{\partial f}{\partial \eta} + \frac{\partial f}{\partial s} + \frac{f}{2s}\right] \qquad [2.7.2]$$

Finally, the momentum equation can be rewritten in terms of the transformed stream function f and its derivatives w.r.t the variable η (denoted as f', f'', etc.), the variable s, and the freestream streamwise velocity u_∞. This process involves numerous intermediate steps to handle the derivative, partial derivative, and double partial derivative terms, and employs the variable and coordinate transformations that have been derived thus far. The transformed momentum equation is seen below:

$$ff'' + f''' + [1 - (f')^2]\frac{2s}{u_\infty}\frac{du_\infty}{ds} = 2s\left[f'\frac{\partial f'}{\partial s} - f''\frac{\partial f}{\partial s}\right] \qquad [2.8]$$

The final simplification step defines the variable β, assumed to be a constant for the purposes of this problem. This assumption also implies that the derivative of f and f' w.r.t. to s are zero, which yields the final expression Eq. [2.9], and is formally known as the Falkner-Skan equation:

$$ff'' + f''' + [1 - (f')^2]\beta = 0 \qquad [2.9]$$

This third order ODE represents the equations that govern the flow of two-dimensional incompressible fluid in such a way where the velocity profile, $f'(\eta)$, does not change along s. This has no analytical solution, nor does the Reynold number appear as a parameter. Only after applying the boundary conditions and using numerical methods can a solution be obtained, after which the transformations can be reversed to determine the boundary layer thickness as a function of the Reynold number.

To determine an analytical solution, the provided MATLAB script 'falkner_skan.m' is used, which finds an analytical solution for Eq. [2.9] that satisfies the boundary conditions for a given β. For a flat plate, $\beta = 0$. The

MATLAB script returns a vector of similarity variable (η) values, and a matrix containing the similarity function ($f(\eta)$), the velocity profile ($f'(\eta)$), and the profile gradient ($f''(\eta)$).

To determine an expression for the boundary layer thickness δ, the definition of the similarity variable η is used. Knowing that the similarity variable is proportionally related to $\frac{y}{\delta}$, and knowing that the similarity variable is as defined as Eq. [2.4], a specific value for η shall define the expression for boundary layer thickness.

To determine this value, let the boundary layer thickness be defined as the distance y from the wall for which boundary layer velocity is 99% that of the freestream velocity. This definition is somewhat arbitrary, however fairly standard. At this point, the boundary layer velocity is nearly indistinguishable from the non-boundary layer flow. Thus, the η value for which $f'(\eta) = \frac{u}{u_\infty} = 0.99$ can be used with Eq. [2.4.1] to derive this expression.

Using MATLAB, the Falkner-Skan solver function is used to generate the output parameters for $\beta = 0$. A for loop is then constructed to iterate over the η output vector, extract the corresponding velocity profile value, and determine if it is equal to or greater than 0.99. The value for η which satisfies this condition is recorded. This value was recorded as $\eta = 3.4722$. Using this similarity variable value and Eq. [4.2.1], an expression for the boundary layer thickness is derived as seen below:

$$\eta_\delta \propto \frac{y_\delta}{\delta} = y\sqrt{\frac{u_\infty}{2vx}} \to \delta = y_\delta = \eta_\delta\sqrt{\frac{2vx}{u_\infty}}$$

$$\delta = \eta_\delta\sqrt{2} \cdot \frac{x}{x}\sqrt{\frac{2vx}{u_\infty}} = \sqrt{2}\eta_\delta x \cdot \sqrt{\frac{vx}{u_\infty x^2}} = \sqrt{2}\eta_\delta x \cdot \sqrt{\frac{v}{u_\infty x}}$$

Knowing that u_∞ is equal to U_∞ here, and the definition of the Reynold number as:

$$Re = \frac{\mathrm{U}_\infty x}{v}$$

Thus, the rightmost square root quantity is simply $\left(\sqrt{Re}\right)^{-1}$, and a final expression for δ is obtained:

$$\delta = \frac{\sqrt{2}\eta_\delta x}{\sqrt{Re}}$$

Solving:

$$\delta = \frac{4.9105x}{\sqrt{Re}} \qquad [(a)]$$

Next, to determine an expression for the shear force at the wall, first consider the known definition of the shear stress in fluid dynamics:

$$\tau = \mu\frac{\partial u}{\partial y}$$

Using the transformation derived in Eq. [2.6.1], the streamwise velocity as defined in Eq. [2.7.1], and the transformed boundary condition at the wall, where $y = 0$, the following expression is obtained:

$$\tau = \frac{\mu u_\infty^2}{\sqrt{2vs}}f''(0)$$

The MATLAB script outputs various values for $f''(0)$ used to converge on a solution in the command window. The final convergence value for $f''(0)$ was determined to be 0.46960. The shear stress is found as the following:

$$\tau = \frac{f''(0)}{\sqrt{2}}\frac{\mu u_\infty^2}{\sqrt{\nu s}} = 0.3321\frac{\mu u_\infty^2}{\sqrt{\nu s}}$$

Next, consider the definition of the aerodynamic friction coefficient:

$$C_f = \frac{\tau}{\frac{1}{2}\rho U_\infty^2}$$

Where the Reynold number is:

$$Re_x = \frac{\mathrm{U}_\infty x}{\nu} = \frac{\mathrm{U}_\infty x\rho}{\mu}$$

Manipulating:

$$\tau = 0.3321\frac{\mu u_\infty^2}{\sqrt{\nu s}} = 0.3321\sqrt{\frac{\mu\rho u_\infty^3}{x}}$$

$$C_f = \frac{2\cdot 0.3321}{\sqrt{Re}} = \frac{0.6641}{\sqrt{Re}} \qquad [(b)]$$

Next, the average skin friction coefficient is found by integrating the friction coefficient along the x dimension, commonly the chord, to obtain an average value. Consider the definition of the average skin friction coefficient, where S_{wet} is the wetted area Lb:

$$\overline{C_f} = \frac{D_f}{q_\infty S_{wet}}$$

In two dimensions, and if L is the plate length in the x dimension, the integration yields the following expression:

$$\overline{C_f} = \frac{\mathrm{b}}{q_\infty x\mathrm{b}}\int_0^x \tau\ dx = \frac{\mathrm{b}}{q_\infty x\mathrm{b}}\int_0^x C_f q_\infty\ dx = \frac{1}{x}\int_0^x C_f\ dx$$

$$\overline{C_f} = \frac{1}{x}\int_0^x C_f\ dx = \frac{1}{x}\int_0^x \frac{0.6641}{\sqrt{Re}}\ dx = \frac{0.6641}{x}\sqrt{\frac{\nu}{\mathrm{U}_\infty}}\int_0^x \frac{1}{\sqrt{x}}\ dx$$

$$\overline{C_f} = \frac{0.6641}{x}\sqrt{\frac{\nu}{\mathrm{U}_\infty}}2\sqrt{x} = \frac{2\cdot 0.6641}{x}\sqrt{\frac{\nu x}{\mathrm{U}_\infty}} = 2\cdot 0.6641\sqrt{\frac{\nu}{u_\infty x}}$$

$$\overline{C_f} = 2\cdot\frac{0.6641}{\sqrt{Re}} = \frac{1.3282}{\sqrt{Re}} \qquad [(c)]$$

To determine an expression for the displacement thickness of a flat-plate boundary layer, defined as the distance by which the streamlines outside of the boundary layer are shifted due to the presence of the boundary layer, and which accounts for the reduction in mass flow due to the boundary layer, consider the following definition of the displacement thickness δ^*:

$$\rho_e u_e \delta^* = \int_0^\delta \rho(u_e - u)\ dy$$

Since we are considering a low speed, incompressible boundary layer, where density is constant, this reduces to:

$$\delta^* = \int_0^\delta \left(1 - \frac{u}{u_e}\right) dy$$

The solution to the Falkner-Skan equation is numerical, and the value for the streamwise velocity in the boundary layer is asymptotically approximated to the defined 0.99% of freestream velocity, and thus the resulting obtained value for δ is somewhat arbitrary in this calculation, if it is equal to or greater than δ. This allows the following expression to be derived, using the transformation derived in Eq. [2.4.2]:

$$\delta^* = \sqrt{\frac{2vx}{u_\mathrm{e}}} \int_0^\infty (1 - f')\ d\eta = \frac{\sqrt{2}x(\eta_\mathrm{e} - f_e)}{\sqrt{Re}}$$

From the MATLAB script, the value for η_e is known to be 3.4722, and the corresponding f_e value is 2.2589. Substituting these values yields the following final expression for the displacement thickness:

$$\delta^* = \frac{\sqrt{2}x(3.4722 - 2.2589)}{\sqrt{Re}} = \frac{1.7159x}{\sqrt{Re}} \qquad [(d)]$$

Results:

$$\delta = \frac{4.9105x}{\sqrt{Re}}$$

$$C_f = \frac{0.6641}{\sqrt{Re}}$$

$$\overline{C_f} = \frac{1.3282}{\sqrt{Re}} = 2 \cdot C_f$$

$$\delta^* = \frac{1.7159x}{\sqrt{Re}}$$

Discussion:

Firstly, the derived expression for the boundary layer thickness δ matches the given expression with less than a two percent margin. This is likely just due to the numerical approximation using MATLAB. This is not to say that the numerical approximation is not accurate, because the derived expression for the total skin friction coefficient $\overline{C_f}$ matches the accepted expression to the given uncertainty, whether rounded or truncated. Since this value is also determined using the same MATLAB function and numerical methods, and the inconsistency is nearly negligible, it is a safe bet that the expressions derived above are accurate to an acceptable value.

From chapter 4 of the textbook, all these expressions are derived numerically, and the answers presented here for all 4 parts of this question also match the ones presented there. The textbook also states that the reasoning behind using the boundary layer displacement, δ^*, is a "… more significant measure of the boundary layer… (pg. 178)", compared to the boundary layer thickness δ. The discrepancy between the derived value above and the expression presented in the textbook is less than 1%, reinforcing the relative accuracy of the numerical methods used for this homework.

The textbook uses a value η of 3.5, whereas the η value used for these derivations is slightly less, at 3.4722. The next iteration of η is 3.5139, which may provide derived expressions that are marginally closer to the accepted values, but the gain in accuracy will likely be negligible, even at ‘small’ Reynold numbers that correspond to laminar flow, $Re \approx\leq 500{,}000$.

2. Question 2:

a) *The Falkner-Skan equation holds only when boundary-layer theory is valid, meaning the flow cannot be separated. Find the maximum negative wedge angle for the Falkner-Skan problem for which the flow remains attached.*

b) *What is the value of the shear stress at the wall for this case? Explain using computed results.*

Assume: Incompressible, Newtonian fluid. Steady, two-dimensional flow. Constant viscosity. Thin boundary layer, no flow separation. No slip condition, no flow penetration at the wall, boundary layer velocity approaches free-stream velocity at outer edge.

Given: MATLAB script 'falkner_skan.m'

Solution:

To determine the maximum negative wedge angle for which the flow remains attached, a condition for which flow detaches must be determined. Flow becomes detached when the fluid particles cannot overcome the adverse pressure gradient that corresponds to cases where the inviscid flow is decelerating ($dp/dx > 0$). This happens commonly on bluff bodies – around a cylinder, or an airfoil at a high angle of attack, or more generally and in this case, a negative wedge angle. This condition results in the boundary layer flow slowing greatly and eventually attempting to flow backwards in a sense, which is not possible due to the steady flow of oncoming fluid. Thus, the fluid separates from the surface and forms turbulent vortices and eddies.

The flow velocity at the surface cannot be used, as the boundary condition defines this as *always* equal to zero. However, the definition of the shear stress at the wall, and its relationship with the boundary layer gradient, $f''(\eta)$, can be used. The shear stress measures the rate at which momentum is transferred from the freestream flow to the wall due to the viscous effects in the boundary layer. An adverse pressure gradient drains energy from and decelerates the boundary layer flow, eventually reducing it to stagnation. Consider again the definition of shear stress:

$$\tau = \mu \frac{\partial u}{\partial y} = \frac{\mu u_\infty^2}{\sqrt{2\nu s}} f''(0)$$

When stagnation or reverse flow in the boundary layer occurs, $\frac{\partial u}{\partial y} \leq 0$. This condition is reflected in the Falkner-Skan equation for when $f''(0) \leq 0$. Thus, the condition for which flow detaches corresponds to the value of β which drives the profile gradient to zero. Also, note that this solution to the profile gradient is unique in the fact that it does not depend on the freestream conditions like velocity or the Reynold number.

To determine this value, the provided MATLAB script for the Falkner-Skan equation is used. First, a general range for the desired β is determined by simply guessing. For $\beta = -0.2$, the code does not converge on a value for $f''(0)$, meaning β must be larger. For $\beta = -0.19$, $f''(0) = 0.08570$. To get a more precise answer, an algorithm was developed to run the function for $-0.20 < \beta < -0.19$ in increments of 0.0001.

Development of this algorithm involved a small change to the function provided. An additional output parameter was added, which returns a logical binary output indicating whether the numerical error is less than the maximum allowable error AND if the infinity value is less than the maximum infinity value. Neither the maximum error nor the maximum infinity values were modified. This output parameter is an indication of whether the numerical methods used to obtain a solution were able to converge on a value, i.e. whether the value of β is valid for these purposes. This was used programmatically to update the range of the beta vector to increasingly more precise values, which significantly reduced the execution time and complexity.

This algorithm determined that the maximum negative value for beta is approximately -0.19883773, where the value for $f''(0)$ converged to 0.00007 for infinity equal to 13. The corresponding wedge angle θ, according to the description written within the provided function, is seen below:

$$\theta = \beta \cdot \frac{\pi}{2} = -17.8953957°$$

The shear stress at the wall for this case was discussed above, but a numerical value can be determined by considering the definition of fluid shear stress:

$$\tau = \mu \frac{\partial u}{\partial y} = \frac{\mu u_\infty^2}{\sqrt{2\nu s}} f''(0)$$

The true shear stress depends on the flow parameters and the fluid, however with such a small value for the velocity gradient, it can be safely approximated to zero:

$$\tau = \frac{\mu u_\infty^2}{\sqrt{2\nu s}} f''(0) = 0.00007 \cdot \frac{\mu u_\infty^2}{\sqrt{2\nu s}} \approx 0$$

Results:

$$\beta = -0.19883773$$
$$f''(0) = 0.00007$$
$$\theta = -17.8953957°$$
$$\tau \approx 0$$

Discussion:

Much of the explanation according to fluid dynamic principles and computed results was included in the solution portion, however a concise summary of these points is here. The first question in this part of the homework validated the Falkner-Skan numerical solution that was provided for our use, which was used here. The difference, however, is that the parameter β here is no longer equal to 0, thus a pressure differential is present. From the initial derivation of the Falkner-Skan presented in question one, it is known that this parameter encapsulates the variable s, which is a function of the free-stream velocity and x coordinate, and partial derivatives of s w.r.t. freestream velocity. Once again referencing the textbook however, the answers derived for this question match those presented in chapter 4, whether rounded or truncated. The power presented by using MATLAB allowed me to find an answer for the wedge angle to more significant digits, but this was probably (definitely) overkill, as changing the angle by a few thousandths or smaller is completely imperceptible and impacts the shear stress at the wall by an imperceptible amount as well. At some point, near zero is just zero.

It is interesting to note that the separation angle is independent of many of the characteristic variables of the flow, meaning that a variation in β and the resulting computations is largely independent of many specific dimensioned parameters that describe the flow.

III. Part 3

1. **Question 1:**

The MQ-9 Reaper is a Remotely Piloted Aircraft designed for long-endurance, high-altitude surveillance.

a) *The MQ-9 design flight condition is for a speed of* $313\ km/hr$ *(*$87\ m/s$*) at an altitude of 25,000 feet. Find the wing parasite drag coefficient,* $C_{D_{P_{wing}}}$*, at this flight condition.*

b) *The parasite drag coefficient for the rest of the aircraft is 0.013. Therefore, the total parasite drag coefficient for the airplane is,* $C_{D_{P_{wing}}} + 0.013$*. Plot the total aircraft viscous drag vs. forward speed at an altitude of 25,000 ft. Use a range of speeds from* $70\ m/s$ *to* $140\ m/s$*.*

c) *On the same plot as for part (b), plot the induced drag vs. forward speed at the nominal cruising weight at an altitude of 25,000 ft. Discuss the relative importance of induced, parasite, and total drag as a function of speed. What appears to be the condition for minimum total drag?*

d) *The NLF 1015 airfoil was designed specifically to maintain laminar flow over much of the surface, even at higher Reynolds numbers. Explain how (and why) this could affect your answer to part (a).*

Assume: Steady, incompressible flow. Thin, flat-plate airfoil. Average wing chord for calculations. Non-interfering drag components. Smooth, non-rough airfoil surface.
Given: Flight condition: $313\ km/hr = 87\ m/s$ at $25{,}000\ ft$. 6% of the wing planform is covered by fuselage. Airfoil is similar to NLF 1015, with 15% maximum thickness.

Nominal Cruising Weight (W)	$43120\ N$
Wing Area (S)	$23.8m^2$
Wing Span (b)	$20\ m$
Wing Sweep (Λ)	0
Oswald Efficiency (e)	0.92
Density (ρ)	$0.5489 kg/m^3$
Viscosity (μ)	$1.54 \cdot E^{-5} kg/m \cdot s$
Speed of Sound (a)	$309\ m/s$

Solution:

To estimate the parasitic drag for an aircraft, some assumptions and simplifications are made. From the given notes and flight parameters, it is seen that the MQ-9 is traveling at sufficiently subsonic speeds to allow the application of the wing and fuselage methods outlined by Shevell (1989). Parasite drag is also referred to as zero-lift drag and refers to the drag experienced by an aircraft when it is not generating any lift, thus no induced drag, and consists primarily of form, friction, and interference drag. The application of the wing method works under the assumption that there is no interference drag, thus the computed quantity is mainly composed of form and frictional drag. Consider the definition of this parasitic drag, according to Shevell:

$$C_{D_P} = \sum_{\mathrm{i=1}}^{\mathrm{N}} \frac{K_i \overline{C_{f_i}} S_{wet_i}}{S_{\mathrm{ref}}} \qquad [3.1]$$

Here, N is the number of components being considered for the aircraft, K is the form factor, $\overline{C_f}$ is the total skin-friction coefficient, S_{wet} is the wetted area, and S_{ref} is the reference area, typically the planform area. Surface roughness correction is omitted for this solution under the assumption that the wing surface is smooth enough that roughness is negligible. To determine the parasite drag for just the wing, N is equal to 1. According to the course textbook, K is found using empirical data as a function of thickness ratio and quarter-chord wing sweep, which is presented on page 266 in Fig. [5.21]. For the given wing parameters, with maximum 15% thickness and 0 degree wing sweep, the value for K is estimated to be around 1.35:

$$\mathrm{K} \approx 1.35$$

The equation for the total skin friction coefficient depends on the type of boundary layer; laminar boundary layers and turbulent boundary layers have different equations. In practice, the boundary layer starts off as laminar, however, as the adverse pressure gradient acts on the boundary layer, it typically transitions to turbulent. The equation for the laminar boundary layer was found in the solution for part (c) in part 2, question 1 of this assignment. The equation for the turbulent boundary layer was not derived in this assignment, however according to the textbook, it is as follows, known to be the Prandtl formulation:

$$\overline{C_f} = \frac{0.074}{(Re_x)^{0.2}}$$

The textbook then states that the more accurate formulation for the total skin friction coefficient is found according to the Prandtl-Schlichting formulation. This formula includes a correction for the transition between boundary layer types, assuming the transition takes place at rough Reynold number of 500,000. This form will be used for part (a) of this assignment, and is given as follows:

$$\overline{C_f} = \frac{0.455}{(log_{10}Re_x)^{2.58}} - \frac{1700}{Re_x} \qquad [3.2]$$

The final equation needed to determine the wing parasite drag is the wetted area. The wetted area can be calculated according to Kroo (2003), and is shown below:

$$S_{wet} = 2.0(1 + 0.2t/c)S_{exposed} \qquad [3.3]$$

This equation accounts for the increase in area due to thickness, and the exposed area is the portion of the planform that is not covered by the fuselage. It is scaled by a factor of two to account for both the upper and lower surfaces of the wing.

The Reynold number must first be found using the typical definition, seen below. The characteristic length to be used here is the mean aerodynamic chord length, to account for any wing taper that may exist:

$$c_{mac} = \frac{S}{b} = 1.19m$$

$$Re_x = \frac{\mathrm{V}_\infty c_{mac}\rho}{\mu} = 3.69 \cdot E^6$$

$$\overline{C_f} = \frac{0.455}{(log_{10}Re_x)^{2.58}} - \frac{1700}{Re_x} = 0.0030809$$

$$S_{exposed} = S_{ref} \cdot (1 - 0.06) = 22.372m^2$$

$$S_{wet} = 2.0(1 + 0.2t/c)S_{exposed} = 46.086m^2$$

Finally, the wing parasite drag is found as the following:

$$C_{D_{P_{wing}}} = \frac{K_i\overline{C_{f_i}}S_{wet_\mathrm{i}}}{S_\mathrm{ref}} = 0.00805389$$

To determine the total viscous drag of the aircraft, the total drag coefficient for the aircraft is assumed to be the sum of the total parasite drag and the induced drag coefficients. The total parasite drag coefficient of the aircraft is given as:

$$C_{D_P} = C_{D_{P_{wing}}} + 0.013 = 0.0211$$

The induced drag coefficient, as derived in previous assignments, is found to be a function of the lift coefficient, the aspect ratio, and the Oswald efficiency factor, as seen below:

$$C_{D_i} = \frac{C_L^2}{\pi eAR}$$

The lift coefficient can be determined as a function of the lifting force, the dynamic pressure, and the wing planform area. This definition is chosen over other derived definitions, as is the simplest definition and circumvents the need to determine root chord circulation, or the effective angle of attack. For the purposes of this solution, the MQ-9 is assumed to be flying under the SLUF condition, where the lift is equal to the given nominal cruising weight. This formulation is seen below:

$$C_L = \frac{L}{q_\infty S} = \frac{\mathrm{W}}{\frac{1}{2}\rho V_\infty^2 S}$$

These equations can be used to determine the total drag coefficient of the aircraft at a given cruising speed under SLUF conditions. Although the wing parasite drag coefficient technically varies as a function of speed, it will be assumed to be a constant and equal to the value found in part (a) for this part of the solution. Thus, the total viscous drag coefficient as a function of cruising speed can be determined as the following, from which the total drag can be calculated. These equations are seen below:

$$C_D = C_{D_P} + C_{D_i}(\mathrm{V}_\infty)$$

$$D = C_D q_\infty S = C_{D\frac{1}{2}}\rho V_\infty^2 S$$

These calculations were performed using MATLAB for $70m/s \leq V_\infty \leq 140m/s$, and the total viscous drag force, the induced drag force, and the parasite drag force was plotted against the airspeed. This plot is shown in the results as Fig. [3.1], and is analyzed in the discussion section.

If the airfoil is designed to maintain laminar flow over much of the surface, then the laminar boundary layer equation for the total skin friction coefficient should be used for the wing. In this case, the value for the total wing skin friction coefficient is found, as shown below:

$$\overline{C_f} = \frac{1.3282}{\sqrt{Re}} = 0.00069142$$

$$C_{D_{P_{wing}}} = \frac{K_i \overline{C_{f_i}} S_{wet_i}}{S_{\text{ref}}} = 0.00180748$$

$$C_{D_P} = C_{D_{P_{wing}}} + 0.013 = 0.0148075$$

It is clear at this point that maintaining a laminar boundary layer reduces the wing parasitic drag significantly by a factor of roughly 4 times, and thus reduces the overall total parasitic drag as well. The same plot generated for parts (b) & (c) was generated for this part as well, seen in the results as Fig. [3.2]. This is also analyzed at length in the discussion section.

Results:

$$C_{D_{P_{wing}}} = 0.00805389$$

$$C_{D_P} = 0.0211$$

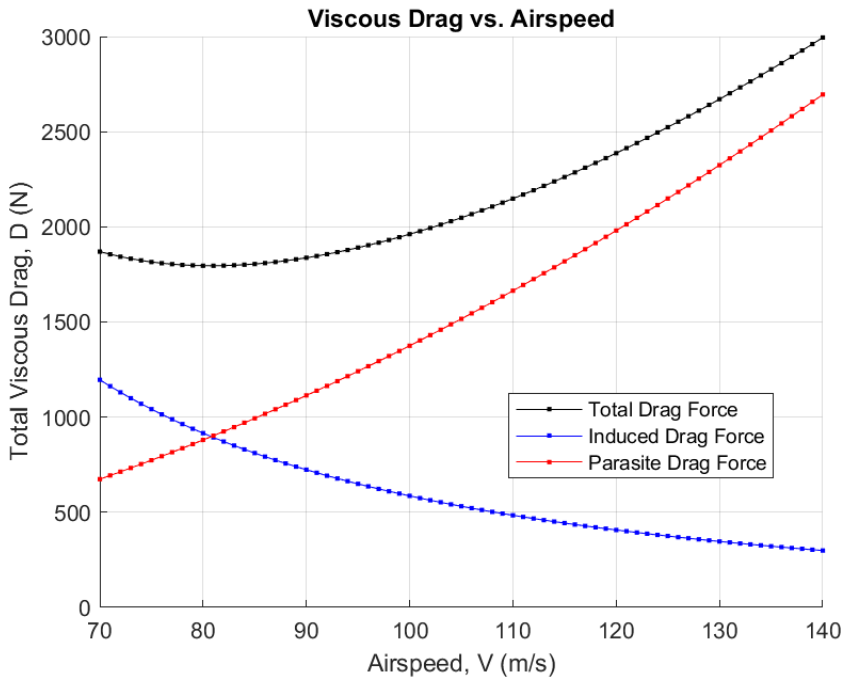

Fig. [3.1]: Total viscous drag, induced drag, and parasite drag for the MQ-9 Reaper cruising at airspeeds between 70 m/s and 140 m/s at 25,000 feet of altitude

$$C_{D_{P_{wing,\text{laminar}}}} = 0.00180748$$

$$C_{D_{P,\text{laminar}}} = 0.0148075$$

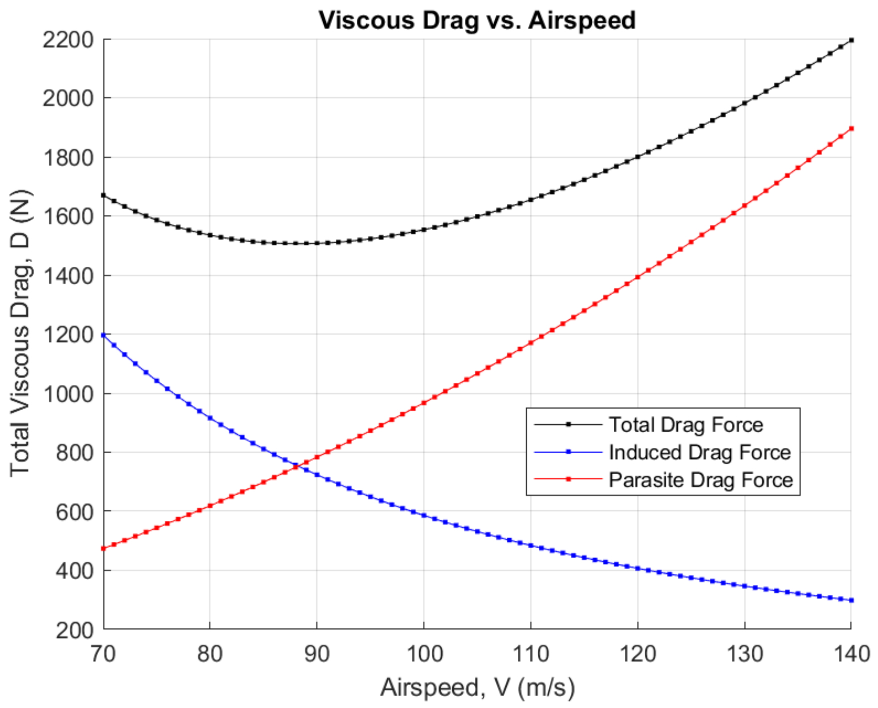

Fig. [3.2]: Total viscous drag, induced drag, and parasite drag for the MQ-9 Reaper cruising at airspeeds between 70 m/s and 140 m/s at 25,000 feet of altitude, assuming a laminar boundary layer

Discussion:

First, from the results computed using the Prandtl-Schlichting formulation, it is shown that the induced drag and parasite drag have inverse relationships with the airspeed. Parasite drag increases as speed increases and does so relatively linearly. This is expected, as a higher flow speed means that the fluid particles have more energy to transfer to the aircraft. The induced drag decreases as velocity increases, which is also expected, as it was derived in previous homework assignments. The nature of the parasite and induced drag's relationship with velocity means that the condition for which total viscous drag force is at a minimum is the airspeed for which these two lines intersect. From MATLAB, the closest intersection point was at a speed of 81 m/s, and an analysis of the total drag force shows that it does in fact reach a minimum value at this airspeed.

Comparing the results between the Prandtl-Schlichting formulation and the Prandtl formulation for laminar flow, it is shown that the airframe experiences a lower amount of parasite drag when flow over the wing surface remains laminar. This makes sense, as the laminar boundary layer flow is lower energy than a mixed or turbulent boundary layer, and thus imparts less frictional energy and shear stress on the wing surface. Because the flight conditions remain the same between the two calculations, the lifting characteristics of the airfoil do not vary as a function of the boundary layer type, or at least the variation is negligible, and the assumptions made to solve this question made it such that the induced drag for both cases is the same. Looking at the graph generated for the laminar condition, it is shown that the intersection point of the two drag components is shifted to the right because of the lower parasite drag, and thus the airspeed for minimum drag is increased. The value for the laminar condition for minimum drag is found to be about $8\ m/s$ faster, at roughly $89\ m/s$.

This is a key consideration for aircraft design, as the airspeed for which an aircraft experiences minimum drag can be the most efficient. This most efficient cruising speed seems to be a key parameter for aircraft engineering and requirement-based design. Knowing that the reduction of skin friction drag has a significant impact on this parameter, it is noted that reduction of this type of drag, and parasite drag overall is key for optimizing aircraft design.

For example, if requirements state the optimal cruising speed to be $95\ m/s$, and the current prototype can be approximated to the results from the mixed boundary layer, an engineer with this knowledge should focus on keeping the boundary layer over the wings in a laminar state, as this change is shown here to provide a significant $8\ m/s$ increase. Then, other considerations such as wing surface roughness (which was not considered in this solution), other exposed surface boundary types and roughness, etc. could be modified to meet the requirement. Obviously, this heavily generalized, and this solution is approximate with many assumptions, but is captures the essence.

Appendix A

```
%% Header
% Author: Zakary Steenhoek
% Date: 05 December 2024
% AEE 360 HW08 P1Q1

clc; clear; %close all;

%% Load Data

% Load data from the text files for both airfoils
dataPath = 'C:\Users\zaste\OneDrive\Documents\Software\MATLAB\AEE360\data\NACA0012\';
figPath = 'C:\Users\zaste\OneDrive\Documents\Software\MATLAB\AEE360\HW08\figures\';
dataFiles = dir(fullfile(dataPath));

% To record the data for each airfoil
allData = struct('deg0', zeros(301,4), 'deg15', zeros(301,4));
allBodyCalcs = zeros(2,7); allBodyCalcs(1,1) = 0; allBodyCalcs(2,1) = 15;
allWindCalcs = zeros(2,7); allWindCalcs(1,1) = 0; allWindCalcs(2,1) = 15;
rad15 = deg2rad(15);

%% Extract Data

% For the number data files
for i = 1:length(dataFiles)
    if contains(dataFiles(i).name, {'_0_', '_15_'})
        if contains(dataFiles(i).name, 'Cf')
            % Build path to the file
            filePath = fullfile(dataPath, dataFiles(i).name);
            dataFile = importdata(filePath);

            % Get the angle of attack
            subs = split(dataFiles(i).name, '_'); alphadeg = subs{end-1};

            % Convert to int -> rad
            alphadeg = str2double(alphadeg);
            alpha = alphadeg*pi/180;

            if alphadeg == 0
                allData.deg0(:, 3) = flipud(dataFile.data(:, 4));
            elseif alphadeg == 15
                allData.deg15(:, 3) = flipud(dataFile.data(:, 4));
            else
                break
            end
        else
            % Build path to the file
            filePath = fullfile(dataPath, dataFiles(i).name);
            dataFile = importdata(filePath);

            % Get the angle of attack
            subs = split(dataFiles(i).name, '_'); alphadeg = subs{end-1};

            % Convert to int -> rad
            alphadeg = str2double(alphadeg);
            alpha = alphadeg*pi/180;

            if alphadeg == 0
                allData.deg0(:, 1) = flipud(dataFile.data(:, 1));
                allData.deg0(:, 2) = flipud(dataFile.data(:, 3));
                allData.deg0(:, 4) = flipud(dataFile.data(:, 4));
            elseif alphadeg == 15
                allData.deg15(:, 1) = flipud(dataFile.data(:, 1));
                allData.deg15(:, 2) = flipud(dataFile.data(:, 3));
                allData.deg15(:, 4) = flipud(dataFile.data(:, 4));
            else
                break
            end
        end
    end
end

%% Coefficient Calculations

% Find force coefficients for x and z
allBodyCalcs(1,2) = trapz(allData.deg0(:, 2), allData.deg0(:, 3));
allBodyCalcs(1,3) = trapz(allData.deg0(:, 1), -allData.deg0(:, 4));
allBodyCalcs(1,4) = allBodyCalcs(1,2) + allBodyCalcs(1,3);
allBodyCalcs(1,5) = trapz(allData.deg0(:, 1), allData.deg0(:, 3));
allBodyCalcs(1,6) = trapz(allData.deg0(:, 2), allData.deg0(:, 4));
allBodyCalcs(1,7) = allBodyCalcs(1,5) + allBodyCalcs(1,6);

% Find force coefficients for x and z
allBodyCalcs(2,2) = trapz(allData.deg15(:, 2), allData.deg15(:, 3));
allBodyCalcs(2,3) = trapz(allData.deg15(:, 1), -allData.deg15(:, 4));
allBodyCalcs(2,4) = allBodyCalcs(2,2) + allBodyCalcs(2,3);
allBodyCalcs(2,5) = trapz(allData.deg15(:, 1), allData.deg15(:, 3));
allBodyCalcs(2,6) = trapz(allData.deg15(:, 2), allData.deg15(:, 4));
allBodyCalcs(2,7) = allBodyCalcs(2,5) + allBodyCalcs(2,6);

% Find lift and drag coefficients
allWindCalcs(1,2) = allBodyCalcs(1,2)*cos(0) - allBodyCalcs(1,5)*sin(0);
allWindCalcs(1,3) = allBodyCalcs(1,3)*cos(0) - allBodyCalcs(1,6)*sin(0);
allWindCalcs(1,4) = allBodyCalcs(1,4)*cos(0) - allBodyCalcs(1,7)*sin(0);
allWindCalcs(1,5) = allBodyCalcs(1,2)*sin(0) + allBodyCalcs(1,5)*cos(0);
allWindCalcs(1,6) = allBodyCalcs(1,3)*sin(0) + allBodyCalcs(1,6)*cos(0);
allWindCalcs(1,7) = allBodyCalcs(1,4)*sin(0) + allBodyCalcs(1,7)*cos(0);

% Find lift and drag coefficients
allWindCalcs(2,2) = allBodyCalcs(2,2)*cos(rad15) - allBodyCalcs(2,5)*sin(rad15);
allWindCalcs(2,3) = allBodyCalcs(2,3)*cos(rad15) - allBodyCalcs(2,6)*sin(rad15);
allWindCalcs(2,4) = allBodyCalcs(2,4)*cos(rad15) - allBodyCalcs(2,7)*sin(rad15);
allWindCalcs(2,5) = allBodyCalcs(2,2)*sin(rad15) + allBodyCalcs(2,5)*cos(rad15);
allWindCalcs(2,6) = allBodyCalcs(2,3)*sin(rad15) + allBodyCalcs(2,6)*cos(rad15);
allWindCalcs(2,7) = allBodyCalcs(2,4)*sin(rad15) + allBodyCalcs(2,7)*cos(rad15);

%% Plot Data

figure(80000); clf; hold on; grid on;
xlabel('Chord (dimless)'); ylabel('Friction Coefficient');
```

```
title('NACA 0012 Friction Distribution, alpha = 0 deg');
n = length(allData.deg0);
plot(allData.deg0(n/2:n,1), allData.deg0(n/2:n,3), 'b', ...
    allData.deg0(1:n/2+1,1), allData.deg0(1:n/2+1,3), 'b--');
% plot(allData.deg0(:,1), allData.deg0(:,3), 'b');
legend('Upper Surface', 'Lower Surface');
xlim([0 1]); ylim([-0.002 0.002]);
hold off;

% q = findobj('type','figure');
% autosave(q, '0012_00deg', figPath, 'png', false);

figure(80015); clf; hold on; grid on;
xlabel('Chord (dimless)'); ylabel('Friction Coefficient');
title('NACA 0012 Friction Distribution, alpha = 15 deg');
n = length(allData.deg0);
plot(allData.deg15(n/2:n,1), allData.deg15(n/2:n,3), 'b', ...
    allData.deg15(1:n/2+1,1), allData.deg15(1:n/2+1,3), 'b--');
% plot(allData.deg15(:,1), allData.deg15(:,3), 'b');
legend('Upper Surface', 'Lower Surface');
xlim([0 1]); ylim([-0.002 0.002]);
hold off;

% q = findobj('type','figure');
% autosave(q, '0012_15deg', figPath, 'png', false);

%% Header
% Author: Zakary Steenhoek
% Date: 05 December 2024
% AEE 360 HW08 P1Q2

clc; clear; %close all;

%% Load Data

% Load data from the text files for both airfoils
dataPath = 'C:\Users\zaste\OneDrive\Documents\Software\MATLAB\AEE360\data\NACA4412\';
figPath = 'C:\Users\zaste\OneDrive\Documents\Software\MATLAB\AEE360\HW08\figures\';
dataFiles = dir(fullfile(dataPath));

% To record the data for each airfoil
allData = struct('deg0', zeros(301,4), 'deg15', zeros(301,4));
allBodyCalcs = zeros(2,7); allBodyCalcs(1,1) = 0; allBodyCalcs(2,1) = 15;
allWindCalcs = zeros(2,7); allWindCalcs(1,1) = 0; allWindCalcs(2,1) = 15;
rad15 = deg2rad(15);

%% Extract Data

% For the number data files
for i = 1:length(dataFiles)
   if contains(dataFiles(i).name, {'_0_', '_15_'})
      if contains(dataFiles(i).name, 'Cf')
         % Build path to the file
         filePath = fullfile(dataPath, dataFiles(i).name);
         dataFile = importdata(filePath);
```

```
            % Get the angle of attack
            subs = split(dataFiles(i).name, '_'); alphadeg = subs{end-1};

            % Convert to int -> rad
            alphadeg = str2double(alphadeg);
            alpha = alphadeg*pi/180;

            if alphadeg == 0
                allData.deg0(:, 3) = flipud(dataFile.data(:, 4));
            elseif alphadeg == 15
                allData.deg15(:, 3) = flipud(dataFile.data(:, 4));
            else
                break
            end
        else
            % Build path to the file
            filePath = fullfile(dataPath, dataFiles(i).name);
            dataFile = importdata(filePath);

            % Get the angle of attack
            subs = split(dataFiles(i).name, '_'); alphadeg = subs{end-1};

            % Convert to int -> rad
            alphadeg = str2double(alphadeg);
            alpha = alphadeg*pi/180;

            if alphadeg == 0
                allData.deg0(:, 1) = flipud(dataFile.data(:, 1));
                allData.deg0(:, 2) = flipud(dataFile.data(:, 3));
                allData.deg0(:, 4) = flipud(dataFile.data(:, 4));
            elseif alphadeg == 15
                allData.deg15(:, 1) = flipud(dataFile.data(:, 1));
                allData.deg15(:, 2) = flipud(dataFile.data(:, 3));
                allData.deg15(:, 4) = flipud(dataFile.data(:, 4));
            else
                break
            end
        end
    end
end

%% Coefficient Calculations

% Find force coefficients for x and z
allBodyCalcs(1,2) = trapz(allData.deg0(:, 2), allData.deg0(:, 3));
allBodyCalcs(1,3) = trapz(allData.deg0(:, 1), -allData.deg0(:, 4));
allBodyCalcs(1,4) = allBodyCalcs(1,2) + allBodyCalcs(1,3);
allBodyCalcs(1,5) = trapz(allData.deg0(:, 1), allData.deg0(:, 3));
allBodyCalcs(1,6) = trapz(allData.deg0(:, 2), allData.deg0(:, 4));
allBodyCalcs(1,7) = allBodyCalcs(1,5) + allBodyCalcs(1,6);

% Find force coefficients for x and z
allBodyCalcs(2,2) = trapz(allData.deg15(:, 2), allData.deg15(:, 3));
allBodyCalcs(2,3) = trapz(allData.deg15(:, 1), -allData.deg15(:, 4));
allBodyCalcs(2,4) = allBodyCalcs(2,2) + allBodyCalcs(2,3);
allBodyCalcs(2,5) = trapz(allData.deg15(:, 1), allData.deg15(:, 3));
allBodyCalcs(2,6) = trapz(allData.deg15(:, 2), allData.deg15(:, 4));
allBodyCalcs(2,7) = allBodyCalcs(2,5) + allBodyCalcs(2,6);

% Find lift and drag coefficients
allWindCalcs(1,2) = allBodyCalcs(1,2)*cos(0) - allBodyCalcs(1,5)*sin(0);
allWindCalcs(1,3) = allBodyCalcs(1,3)*cos(0) - allBodyCalcs(1,6)*sin(0);
allWindCalcs(1,4) = allBodyCalcs(1,4)*cos(0) - allBodyCalcs(1,7)*sin(0);
allWindCalcs(1,5) = allBodyCalcs(1,2)*sin(0) + allBodyCalcs(1,5)*cos(0);
allWindCalcs(1,6) = allBodyCalcs(1,3)*sin(0) + allBodyCalcs(1,6)*cos(0);
allWindCalcs(1,7) = allBodyCalcs(1,4)*sin(0) + allBodyCalcs(1,7)*cos(0);

% Find lift and drag coefficients
allWindCalcs(2,2) = allBodyCalcs(2,2)*cos(rad15) - allBodyCalcs(2,5)*sin(rad15);
allWindCalcs(2,3) = allBodyCalcs(2,3)*cos(rad15) - allBodyCalcs(2,6)*sin(rad15);
allWindCalcs(2,4) = allBodyCalcs(2,4)*cos(rad15) - allBodyCalcs(2,7)*sin(rad15);
allWindCalcs(2,5) = allBodyCalcs(2,2)*sin(rad15) + allBodyCalcs(2,5)*cos(rad15);
allWindCalcs(2,6) = allBodyCalcs(2,3)*sin(rad15) + allBodyCalcs(2,6)*cos(rad15);
allWindCalcs(2,7) = allBodyCalcs(2,4)*sin(rad15) + allBodyCalcs(2,7)*cos(rad15);

%% Plot Data

figure(84400); clf; hold on; grid on;
xlabel('Chord (dimless)'); ylabel('Friction Coefficient');
title('NACA 4412 Friction Distribution, alpha = 0 deg');
n = length(allData.deg0);
plot(allData.deg0(n/2:n,1), allData.deg0(n/2:n,3), 'b', ...
    allData.deg0(1:n/2+1,1), allData.deg0(1:n/2+1,3), 'b--');
% plot(allData.deg0(:,1), allData.deg0(:,3), 'b');
legend('Upper Surface', 'Lower Surface');
xlim([0 1]); ylim([-0.002 0.002]);
hold off;

q = findobj('type','figure');
autosave(q, '4412_00deg', figPath, 'jpg', false);

figure(84415); clf; hold on; grid on;
xlabel('Chord (dimless)'); ylabel('Friction Coefficient');
title('NACA 4412 Friction Distribution, alpha = 15 deg');
n = length(allData.deg0);
plot(allData.deg15(n/2:n,1), allData.deg15(n/2:n,3), 'b', ...
    allData.deg15(1:n/2+1,1), allData.deg15(1:n/2+1,3), 'b--');
% plot(allData.deg15(:,1), allData.deg15(:,3), 'b');
legend('Upper Surface', 'Lower Surface');
xlim([0 1]); ylim([-0.002 0.002]);
hold off;

q = findobj('type','figure');
autosave(q, '4412_15deg', figPath, 'jpg', false);

%% Header
% Author: Zakary Steenhoek
% Date: 05 December 2024
% AEE 360 HW08 P2Q1

clc; clear; clf; % close all;
```

```
%% Falkner Skan

beta = -0.19883773;
theta = beta/2*180;

[eta, f, conv] = falkner_skan(beta);

for i = 1:numel(eta)
    profile = f(2,i);

    if (profile >= 0.99)
        sol = eta(i);
        break
    end
end

if conv
    fprintf('true\n')
end


sol*sqrt(2);
f(3,1)*sqrt(2)^-1;
(sol-f(1,i))*sqrt(2);

%% Header
% Author: Zakary Steenhoek
% Date: 05 December 2024
% AEE 360 HW08 P2Q2

clc; clear; clf; % close all;

%% Falkner Skan

% beta = -0.19883;

% Define initial beta range
sol = false;
beta = -0.2+0.001:0.001:-0.19;

% For each beta
for i = 1:numel(beta)

    % Try and find a solution that converges
    [eta, f, conv] = falkner_skan(beta(i));

    if conv
        % Redefine more precise beta
        beta = beta(i-1)+0.0001:0.0001:beta(i);

        % For each beta
        for ii = 1:numel(beta)

            % Try and find a solution that converges
            [eta, f, conv] = falkner_skan(beta(ii));
```

```
if conv
    % Redefine more precise beta
    beta = beta(ii-1)+0.00001:0.00001:beta(ii);

    % For each beta
    for iii = 1:numel(beta)

        % Try and find a solution that converges
        [eta, f, conv] = falkner_skan(beta(iii));

        if conv
            % Redefine more precise beta
            beta = beta(iii-1)+0.000001:0.000001:beta(iii);

            % For each beta
            for iiii = 1:numel(beta)

                % Try and find a solution that converges
                [eta, f, conv] = falkner_skan(beta(iiii));

                if conv
                    % Redefine more precise beta
                    beta = beta(iiii-1)+0.0000001:0.0000001:beta(iiii);

                    % For each beta
                    for iiiii = 1:numel(beta)

                        % Try and find a solution that converges
                        [eta, f, conv] = falkner_skan(beta(iiiii));

                        if conv
                            % Redefine more precise beta
                            beta = beta(iiiii-1)+0.00000001:0.00000001:beta(iiiii);

                            % For each beta
                            for iiiiii = 1:numel(beta)

                                % Try and find a solution that converges
                                [eta, f, conv] = falkner_skan(beta(iiiiii));

                                if conv
                                    % Redefine more precise beta
                                    betaBest = beta(iiiiii); sol = true; break
                                end
                            end
                        end
                        if sol
                            break
                        end
                    end
                end
                if sol
                    break
                end
            end
        end
```

```
                    if sol
                        break
                    end
                end
            end
            if sol
                break
            end
        end
    end
    if sol
        break
    end
end

%% Header
% Author: Zakary Steenhoek
% Date: 05 December 2024
% AEE 360 HW08 P3

clc; clear;

%% Known Parameters

W = 43120;
S = 23.8;
b = 20;
e = 0.92;
rho = 0.5489;
mu = 1.54e-5;
a = 309;
U = 87;
K = 1.35;
c = 1.19;
Re = 3.69e6;
Sexp = 22.372;
Swet = 46.086;
% CDpw = 0.00805389;
CDpw = 0.00180748;
CDpr = 0.013;
V = 70:1:140;
AR = b^2/S;

CDp = CDpr + CDpw;
q = 0.5.*rho.*V.^2;
CL = W./(q.*S);
CDi = (CL.^2)/(pi*e*AR);
CD = CDi + CDp;
D = CD.*q.*S;
Di = CDi.*q.*S;
Dp = CDp.*q.*S;


figure(1); clf; hold on; grid on;
title('Viscous Drag vs. Airspeed');
xlabel('Airspeed, V (m/s)'); ylabel('Total Viscous Drag, D (N)');
```

```
% xlim([]); ylim([]);
plot(V,D,'k.-')
plot(V,Di,'b.-')
plot(V,Dp,'r.-');
legend('Total Drag Force', 'Induced Drag Force', 'Parasite Drag Force', ...
    'Location','best');
hold off;

[D_min, idx] = min(D);
V_eff = V(idx);

% figPath = 'C:\Users\zaste\OneDrive\Documents\Software\MATLAB\AEE360\HW08\figures\';
% q = findobj('type','figure');
% autosave(q, 'drag_vs_speed_avg', figPath, 'png', false);

function [eta, f, conv] = falkner_skan(beta)
%
%   Falkner-Skan BVPs arise from similarity solutions of viscous,
%   incompressible, laminar flow over a wedge:
%
%         f''' + f*f'' + beta*(1-(f')^2) = 0
%   with f(0) = 0, f'(0) = 0, f'(infinity) = 1 .
%
%   The parameter beta determines the wedge angle, where theta = pi*beta/2.
%   beta = 0 is the limiting case of a flat plate.
%
%   The BVP is solved by imposing the boundary condition at infinity
%   at a finite point 'infinity'. Continuation in this end point is
%   used to get convergence for large values of 'infinity' and to gain
%   confidence from consistent results that 'infinity' is big enough.
%   The solution for one value of 'infinity' is extended to a guess for
%   a bigger 'infinity' using BVPXTEND.
%
%   By default, this example uses the BVP4C solver.
%
%   This function is based on the MATLAB-supplied example fsbvp.m by:
%   Jacek Kierzenka and Lawrence F. Shampine
%   Copyright 1984-2014 The MathWorks, Inc.

global sol

solver = 'bvp4c';

bvpsolver = fcnchk(solver);

% Problem parameter, shared with nested functions.

infinity = 3; maxinfinity = 15;
maxerror = 1e-5; error = maxerror + 1;

options = bvpset('RelTol',1e-6,'AbsTol',1e-9); % changed tolerances in order to get a smooth solution near the wall

% Initial guess satisfying the boundary conditions
%
solinit = bvpinit(linspace(0,infinity,5),[0 1 1]);
sol = bvpsolver(@fsode,@fsbc,solinit,options);
```

```
eta = sol.x;
f = sol.y;
test = f(3,1);
fprintf('Value of f''''(0) computed using infinity = %g is %7.5f.\n',infinity,f(3,1));

while error > maxerror && infinity < maxinfinity
  infinity = infinity + 1;

  solinit = bvpxtend(sol,infinity);   % Extend the solution to the new infinity to use as initial guess for next case.
  sol = bvpsolver(@fsode,@fsbc,solinit,options);
  eta = sol.x;
  f = sol.y;

  fprintf('Value of f''''(0) computed using infinity = %g is %7.5f.\n',infinity,f(3,1));
  error = abs((f(3,1)-test)/f(3,1));
  test = f(3,1);
end
% hold off
conv = error < maxerror && infinity < maxinfinity;

plot(f(2,:),eta);grid on; hold on;

title(['Boundary Layer Velocity Profile, Wedge angle = ',sprintf('%5.2f',beta/2*180),' degrees']);
xlabel('u/Ue'); ylabel('\eta');
% ---------------------------------------------------------------------
% Nested functions -- beta is provided by the outer function.
% The third-order ODE becomes three first-order ODE's

  function dfdeta = fsode(eta,f)
    dfdeta = [ f(2)
      f(3)
       -f(1)*f(3) - beta*(1 - f(2)^2) ];
  end
% ---------------------------------------------------------------------
% The boundary conditions are specified as residuals to be driven to zero
  function res = fsbc(f0,finf)
    res = [f0(1)
      f0(2)
      finf(2) - 1];
  end
% ---------------------------------------------------------------------

end
```