Project SA1—Aircraft Wing Analysis

Summary

The advent of high-performance computing has radically changed the aerospace industry's approach to wing design. In the past, wing sections were based closely on one of the wide range of standard geometries for which experimental data were already available, and optimisation was via extensive wind tunnel testing. Now, most initial section design is based on numerical calculations, with experimental work appearing later in the process. The advantage of this combined approach is that the experimental data which is the backbone of any development project is still obtained, but expensive wind tunnel tests can be targeted on the most promising designs identified by the (relatively) cheap computations. This computer-based project provides an introduction to the numerical design process, in the context of two-dimensional aerofoil sections. Programming experience above and beyond Part I computing coursework activities is NOT necessary.

Introduction

Aims

- To write a Matlab code that calculates the lift and drag on a 2D aerofoil section;
- to design high-efficiency aerofoil sections, using numerical calculations to guide the process;
- to gain an understanding of the aerodynamics of aerofoils, in particular the role of the boundary layer in limiting performance;
- to obtain an appreciation of the strengths and weaknesses of CFD itself.

Activities

Week 1: Write a 2D potential flow panel method. Validate via comparison with analytical solutions for standard test cases. First interim report.

Week 2: Write an integral boundary layer equation solver. Validate via comparison with theoretical and empirical results for laminar and turbulent boundary layers. Second interim report.

Weeks 3 and 4: Combine the potential flow and boundary layer routines to produce an aerofoil analysis code. Design your own 2D aerofoil sections using calculation results to guide the process. Final report.

Assessment

Interim Report 1 (15 marks):	Wednesday, 21st May, 4pm
Interim Report 2 (15 marks):	Wednesday, 28th May, 4pm
Final Report (50 marks):	**Thursday, 12th June, 4pm**

Organisation

In each week, eight hours are timetabled in the DPO. The nominal weekly load for a IIA project is 20 hours/week; you should therefore allow for significant additional work—on code planning, background reading and reporting—outside timetabled sessions.

The DPO sessions are Thursdays 11am-1pm and Mondays 9am-11am & 2pm-6pm. Demonstrators will be available for both morning sessions, and from 2.15 to 4.15 in the afternoon. Attendance at the morning sessions is mandatory, and one mark will be deducted per hour's absence. The allocated afternoon slot is not compulsory, but you should note that you cannot call on demonstrator assistance at other times. The project is designed for participants to work in pairs. Please make sure to form your group in the first session and register by entering your **partner's** CRSid in the Moodle assignment 'Group Registration'.

The two interim reports are joint and shall be submitted online before the first project session of the following week. Late reports will be penalised at 3 marks/day. The final report is individual and is due by **4pm** on **Thursday, 12th June**. Due to the tight Part IIA examining timetable, any reports submitted after this deadline cannot be marked, and will receive no credit. Reports should be **submitted online via the SA1 - Wing Analysis Moodle site**, which can be found at https://www.vle.cam.ac.uk/course/view.php?id=70311.

Getting Going

Matlab can be launched in the DPO either from Linux or Windows. If using Windows, group partners are advised to create a shared folder in Google Drive or One Drive - using their CRSid@cam credentials for authentication - where they can access and edit their common project files. Under Linux you will need to do this via a web browser. On a general note, developing and debugging code tends to be much more efficient when two partners have their eyes on the code line as it is being written/modified, rather than when they develop different

parts of the code separately. It is a good idea to get into the habit of working in a common folder, rather than your personal one, as soon as it becomes possible. In any case, code submitted for assessment will be treated as the joint work of both partners.

You will be guided through the Week 1 and Week 2 tasks by completing a succession of specific exercises. Each will require you to write a Matlab script (select 'New→Script' from the Matlab Editor's 'File' menu to begin one), which should be retained for the duration of the project. Exercise 1 of Week 1 awaits …

Week 1: The Panel Method

Exercise 1: Point vortex stream function

Statement of exercise: The stream function ψ at (x, y) due to a point vortex of circulation Γ, located at (x_c, y_c), is given by

$$\psi(x,y) = -\frac{\Gamma}{4\pi}\log r^2,$$

where

$$r^2 = (x - x_c)^2 + (y - y_c)^2$$

Plot contours of ψ in the region $-2.5 \le x \le 2.5,\ -2 \le y \le 2$, for $x_c = 0.75$, $y_c = 0.5,\ \Gamma = 3.0$.

Step (i): write a Matlab function `psipv.m` that returns the streamfunction value for a single (x,y) location. The first line of the function file should be:

```
function psixy = psipv(xc,yc,Gamma,x,y)
```

Note that the natural logarithm function in Matlab is `log( )`.

Step (ii): Write a script to generate matrices `xm, ym, psi` whose (i,j)-th entries give the values of x, y, and ψ at the (i,j)-th grid point, i.e.

```
xm(i,j) = xmin + (i-1)*(xmax-xmin)/(nx-1);
ym(i,j) = ymin + (j-1)*(ymax-ymin)/(ny-1);
psi(i,j) = psipv(xc,yc,Gamma,xm(i,j),ym(i,j));
```

The parameters `xmin, xmax, nx, ymin, ymax, ny, xc, yc, Gamma` are best set at the beginning of the script. Use `nx = 51` and `ny = 41`.

N.B. It's a good idea to include the commands `clear` and `close all` at the beginning of this and subsequent scripts. The first deletes all the variables lying around in the workspace from whatever you did last, and the second removes all previously generated figures.

Step (iii): Add a call to Matlab's contouring routine at the end of the script:

```
c = -0.4:0.2:1.2
contour(xm,ym,psi,c)
```

Assessment materials: (a) listing of `psipv.m`,
(b) listing of script,
(c) contour plot.

Exercise 2: Reference vortex-sheet-panel stream function

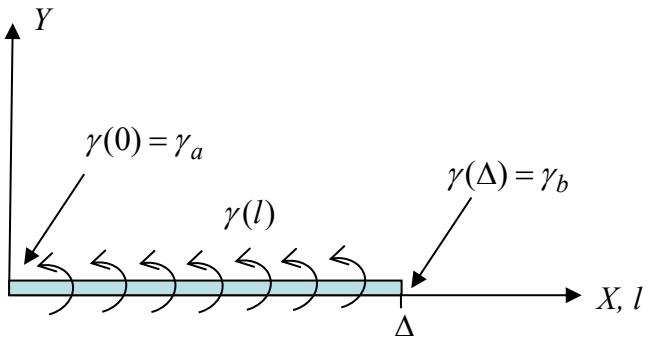

Figure 1. A vortex-sheet panel, of size Δ, with sheet strength varying linearly from γ_a to γ_b.

The stream function ψ at (X,Y) due to the linear vortex-sheet panel shown in Fig. 1 is given by

$$\psi(X,Y)=\gamma_a\left[\left(1-\frac{X}{\Delta}\right)I_0-\frac{I_1}{\Delta}\right]+\gamma_b\left[\frac{X}{\Delta}I_0+\frac{I_1}{\Delta}\right], \qquad (1)$$

where

$$I_0 = -\frac{1}{4\pi}\int_0^{\Delta} \log\left[(l-X)^2 + Y^2\right] dl$$

$$= -\frac{1}{4\pi}\left\{ X\log\left[X^2+Y^2\right] - (X-\Delta)\log\left[(X-\Delta)^2+Y^2\right] - 2\Delta + 2Y\left[\tan^{-1}\left(\frac{X}{Y}\right) - \tan^{-1}\left(\frac{X-\Delta}{Y}\right)\right]\right\} \tag{2}$$

and

$$I_1 = -\frac{1}{4\pi}\int_0^{\Delta} (l-X)\log\left[(l-X)^2 + Y^2\right] dl$$

$$= \frac{1}{8\pi}\left\{\left[X^2+Y^2\right]\log\left[X^2+Y^2\right] - \left[(X-\Delta)^2+Y^2\right]\log\left[(X-\Delta)^2+Y^2\right] - 2X\Delta + \Delta^2\right\}. \tag{3}$$

Statement of exercise: Equation (1) can be written as $\psi(X,Y) = \gamma_a f_a + \gamma_b f_b$, where f_a and f_b are known as 'influence coefficients'. Plot contours of these coefficients in the region $-2.5 \le x \le 2.5,\ -2 \le y \le 2$, for $\Delta = 1.5$. Check your result by generating the corresponding stream functions for the discretised, multiple-point-vortex approximation to the panel shown in Fig. 2.

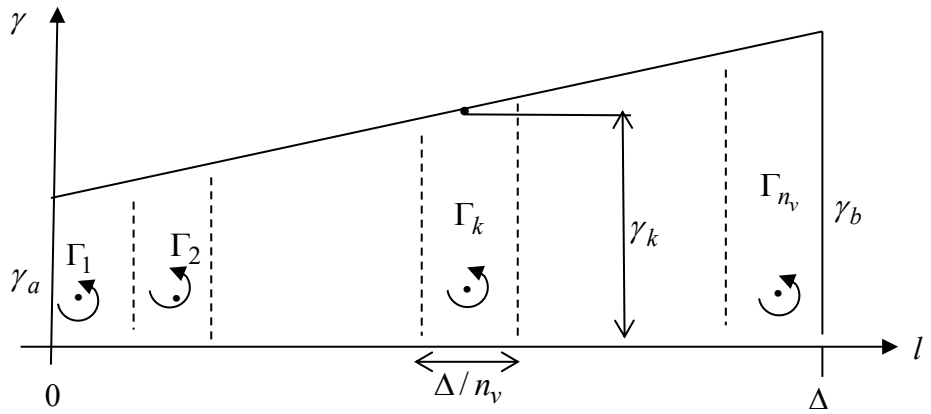

Figure 2. Discretised representation of the linear vortex-sheet panel as a finite set of point vortices.

Step (i): write a Matlab function `refpaninf.m` that returns the influence coefficients for a single (*X,Y*) location. The first line of the function file should be:

```
function [infa infb] = refpaninf(del,X,Yin)
```

Here, `infa` and `infb` are f_a and f_b, and `del` is the value of Δ. The input y variable is `Yin` rather than `Y` because of numerical difficulties in (2) and (3) at the panel boundaries; these can be avoided by setting the following trap:

```
if abs(Yin) < 1e-5
     Y = 1e-5;
else
     Y = Yin;
end
```

The inverse tangent function in Matlab is `atan( )`. You may also find yourself needing to write expressions that span multiple lines in your file. Put the continuation marker … at the end of the line to tell Matlab that the expression continues on the following one.

Step (ii): write a script to generate matrices of x, y and influence coefficient values on a uniform grid, as in Example 1. Use the same number of grid points and set `del = 1.5`.

Step (iii): produce contour plots of `infa` and `infb`, with contour levels defined by the vector `c = -0.15:0.05:0.15`. (To avoid replacing the first plot with the second, bring up another figure window with the command `figure(2)` before contouring `infb`.)

Step (iv): edit your script to include the discretised panel estimates of the stream function at each grid point. This will entail looping over all the point vortices shown in Fig. 2, and adding the individual contributions calculated by `psipv` together to give the overall panel contribution. Use $n_v = 100$ vortices.

Step (v): generate contour plots of the approximated influence coefficients, for comparison with your `refpaninf` results.

Assessment materials: (a) listing of `refpaninf.m`,
(b) listing of script,
(c) four contour plots, showing exact and approximate values of f_a, f_b.

Exercise 3: General vortex-sheet-panel stream function

Statement of exercise: The linear vortex-sheet panel shown in Fig. 3 has a general location specified by its edge coordinates, (x_a,y_a) and (x_b,y_b). For the case $(x_a,y_a) = (4.1,1.3)$, $(x_b,y_b) = (2.2,2.9)$, plot contours of the influence coefficients in the region $0 \le x \le 5,\ 0 \le y \le 4$. Compare your plots against the corresponding results for a discretised, multiple-point-vortex representation of the panel, with the same resolution as you used in Example 2.

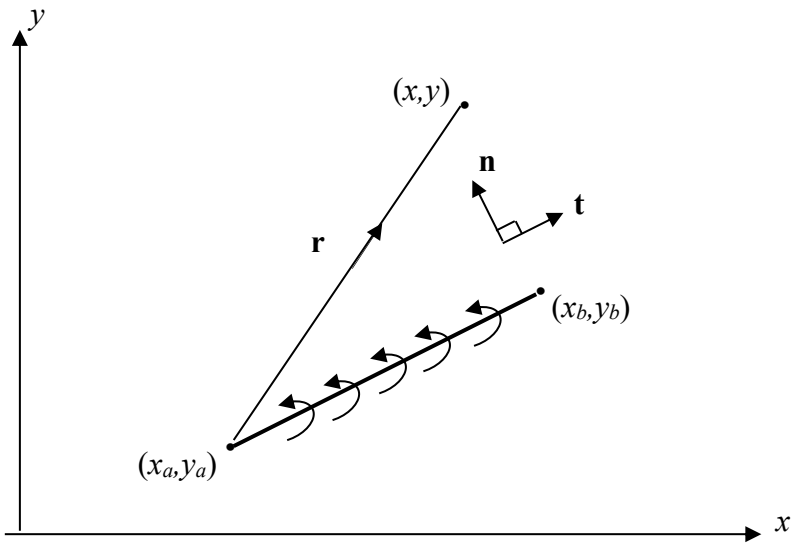

Figure 3. A general linear vortex-sheet panel, with location specified by its edge coordinates.

Step (i): write the Matlab routine defined by

```
function [infa infb] = panelinf(xa,ya,xb,yb,x,y)
```

This routine is to return the influence coefficients f_a, f_b at (x,y) due to our general panel. The first stage is to work out the coordinates of (x,y) in the standard panel reference frame. Calculate the unit vectors tangential and normal to the panel, **t** and **n**. Then evaluate $X = \mathbf{r.t}$, $Y = \mathbf{r.n}$. The desired coefficients now follow straightforwardly from the second stage: a call to `refpaninf`.

Step (ii): edit (a copy of) the script you used for Exercise 2 to generate the required contour plots.

Assessment materials: (a) listing of `panelinf.m`,
(b) listing of script,
(c) four contour plots, showing exact and approximate values of f_a, f_b.

Exercise 4: Cylinder flow streamlines

Statement of exercise: Plot streamlines of the flow past a cylinder of unit radius, in the region $-2.5 \le x \le 2.5,\ -2 \le y \le 2$.

Step (i): define the cylinder panels. For `np` panels, you can generate a set of equispaced angles with `theta = (0:np)*2*pi/np` and then use the formulae `xs(i) = cos(theta(i))`, `ys(i) = sin(theta(i))` to create arrays of x and y coordinates. (Note that the point (1,0) should be repeated as the first and `(np+1)`'th elements of these arrays.) For this exercise use `np = 100`.

Step (ii): define the vortex sheet strength at the panel edges, using $\gamma = -2\sin\theta$. (This corresponds to the cylinder surface velocity for the case of a unit free-stream velocity in the positive x direction.)

Step (iii): loop over x and y locations (indexed as previously, with the same grid resolution), setting the free-stream contribution to the stream function:

```
psi(i,j) = ym(i,j);
```

Step (iv): include a further, nested loop to *add* each panel's contribution to the streamfunction, $\gamma_a f_a + \gamma_b f_b$, using `panelinf` to provide the influence coefficients.

Step (v): plot contours of your completed streamfunction at levels defined by the array `c = -1.75:0.25:1.75`. You can also, if you wish, add the cylinder as follows:

```
hold on
plot(xs,ys)
hold off
```

Here the `hold` command tells Matlab to overlay the new plot on the current figure, rather than replace it.

Assessment materials: (a) listing of script,
(b) streamline plot.

Exercise 5: Panel method solution for the cylinder flow

Statement of exercise: Implement a panel method and solve for the flow past a unit radius cylinder at incidences $\alpha = 0$ and $\alpha = 0.1$, subject to the condition that the surface vortex sheet has zero strength at the point (1,0).

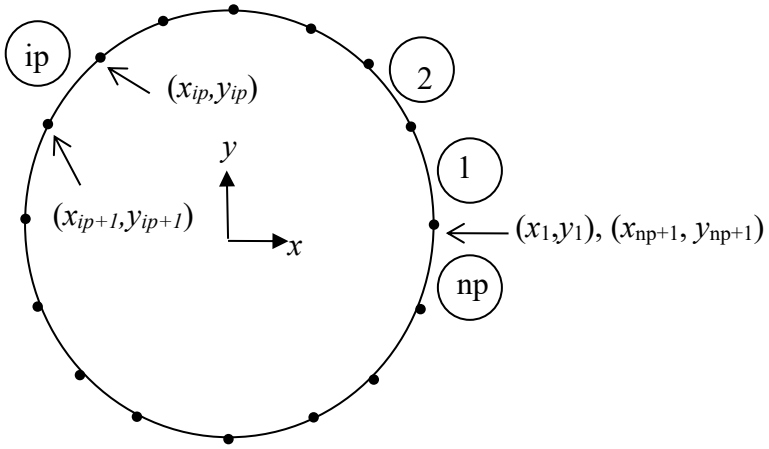

Figure 4. Discretisation of cylinder geometry into `np` equal size panels. Panel numbers are circled.

Panel method formulation

First, observe the nomenclature specified in Fig. 4. The panels are numbered anti-clockwise from the 'trailing edge', (1,0). The `ip`-th panel is defined by the `ip`-th and `(ip+1)`-th coordinate pair, (x_{ip}, y_{ip}) and (x_{ip+1}, y_{ip+1}).

The contribution to the stream function from this panel is $\gamma_{ip} f_a^{(ip)} + \gamma_{ip+1} f_b^{(ip)}$, where the influence coefficients now have a superscript to indicate their associated panel, while γ_a and γ_b have been replaced by γ_{ip} and γ_{ip+1}. The contribution of all the panels to the stream function at one of the panel edges, (x_i,y_i), is thus

$$\psi_p(x_i,y_i) = \sum_{ip=1}^{np} \left[\gamma_{ip} f_a^{(ip)}(x_i,y_i) + \gamma_{ip+1} f_b^{(ip)}(x_i,y_i) \right].$$

Note that, as for the surface coordinates, the trailing-edge value for the vortex sheet strength, γ_1, is repeated as γ_{np+1}. We can rearrange the above expression as follows:

$$\psi_p(x_i,y_i) = \sum_{j=1}^{np+1} \left[\Psi_p \right]_{i,j} \gamma_j , \qquad (4)$$

where

$$\begin{aligned}\left[\boldsymbol{\Psi}_p\right]_{i,j} &= f_a^{(j)}(x_i, y_i), && j = 1;\\ &= f_a^{(j)}(x_i, y_i) + f_b^{(j-1)}(x_i, y_i), && j = 2,3,\ldots, np;\\ &= f_b^{(j-1)}(x_i, y_i), && j = np+1\end{aligned} \tag{5}$$

This alternative formulation expresses $\psi_p(x_i, y_i)$ as the product of the *i*-th row of a matrix $\boldsymbol{\Psi}_p$ with the column vector of vortex sheet strength values. Finally, the overall stream function ψ is given by $\psi_p + \psi_{fs}$ where ψ_{fs} is the free stream contribution.

We can now assemble the set of equations required to solve for the unknown γ_js. The condition that the cylinder surface is a streamline requires the stream function to be constant from panel edge to panel edge, i.e.

$$\psi(x_{i+1}, y_{i+1}) - \psi(x_i, y_i) = 0,$$

for *i* between 1 and $np-1$ (why can't we use the last panel as well?). Splitting the stream function into its (unknown) panel component and its (known) free-stream component, and using (4), we have

$$\sum_{j=1}^{np+1} \left\{ \left[\boldsymbol{\Psi}_p\right]_{i+1,j} - \left[\boldsymbol{\Psi}_p\right]_{i,j} \right\} \gamma_j = \psi_{fs}(x_i, y_i) - \psi_{fs}(x_{i+1}, y_{i+1}), \tag{6}$$

This gives us $np-1$ equations for the $np + 1$ unknown γ_js. The final equations are supplied by the repetition $\gamma_{np+1} = \gamma_1$, and by the additional requirement on the vortex sheet strength at (1,0), i.e. $\gamma_1 = 0$ (once we start working with aerofoil geometries this will become our Kutta condition). We thus complete our system of equations with two auxiliary relations:

$$\gamma_1 = 0, \tag{7}$$

$$\gamma_{np+1} = 0. \tag{8}$$

Equations (6)–(8) can be written as

$$\mathbf{A}\boldsymbol{\gamma} = \mathbf{b}, \tag{9}$$

where **A** is an (np+1) × (np+1) matrix, $\boldsymbol{\gamma}$ is the vector of unknown γ_{ip}s, and **b** contains the stream function contributions. The solution is

$$\boldsymbol{\gamma} = \mathbf{A}^{-1}\mathbf{b}.$$

Step (i): write the Matlab routine

```
function lhsmat = build_lhs(xs,ys)
```

to assemble the matrix **A**. The first stage in this routine is to identify the number of panels and initialize the intermediate matrix $\mathbf{\Psi}_p$:

```
np = length(xs) - 1;
psip = zeros(np,np+1);
```

Next, loop over the panel edge points, each of which corresponds to a row of $\mathbf{\Psi}_p$, and fill the rows according to equations (5). Note that this will entail an inner loop, over panels, inside which you will need to call `panelinf` to obtain the influence coefficients needed to augment the appropriate entries in the matrix.

You can now build **A**. Again, it should first be initialised:

```
lhsmat = zeros(np+1,np+1);
```

If you can't see how to define its elements, go back to (9) and make sure you understand, in detail, how it follows from (6)–(8).

Step (ii): Write the Matlab routine

```
function rhsvec = build_rhs(xs,ys,alpha)
```

to assemble the vector **b** in (9). You should first initialize **b** similarly to **A** in step (i), to ensure that you return a column, rather than row, vector. The entries then follow from (6), (7) and (8). Note that the stream function for the oncoming (unit) flow at incidence α is

$$\psi_{fs}(x,y) = y\cos\alpha - x\sin\alpha\,.$$

Step (iii): write a script to perform the calculations. Your code from Exercise 4 will provide a useful starting point here. Once the flow incidence and cylinder geometry are defined, the commands

```
A = build_lhs(xs,ys);
b = build_rhs(xs,ys,alpha);
gam = A\b;
```

will solve the equations.

Step (iv): add an output stage to your script. This should consist of a calculation of the total cylinder circulation and a plot of the surface velocity (i.e. `gam`) against θ/π. Use the command `axis([0 2 -2.5 2.5])` to ensure that your plots are on the same scale.

Assessment materials: (a) listings of `build_lhs.m` and `build_rhs.m`,
(b) listing of script,
(c) plots of surface velocities for the two requested values of α,
(d) value of circulation for the non-zero value of α.

Reporting

Your first interim report, on the work in Week 1, is due at **4pm on Wednesday, 21st May**. It should consist solely of the assessment materials for Exercises 1–5, in order. If your code structure involves auxiliary functions, include listings of these too. Only one report per project group is required. Please try to present your figures with a decent resolution. If you are preparing your report with a word processor, try .png (Matlab command `print -dpng` *`name`*). If you are using LaTeX, output to .eps (`print -deps2c` *`name`*).

When your report is ready, add your cover sheet to the document and submit it under the 'Interim Report 1' heading. Only one group member need do this, but *both* must approve the report for submission.

Additional Work for Week 1

Get going on some background reading, especially anything that will help for the aerofoil design work that you will be doing in Weeks 3 and 4. The Appendix and the 'Aerofoil section characteristics and design' references are good starting points, but don't stop there; you should be able to find lots of other relevant material, both in print and online.

Week 2: The Boundary-Layer Solver

Exercise 1: Zero-pressure-gradient, laminar boundary layer

Statement of exercise: Write a script to implement Thwaites' solution of the momentum integral equation:

$$\left[\frac{\theta(x)}{L}\right]^2 = \frac{0.45}{Re_L}\left[\frac{u_e(x)}{U}\right]^{-6}\int_0^{x/L}\left[\frac{u_e(x')}{U}\right]^5 d\left(\frac{x'}{L}\right), \tag{10}$$

where u_e is the velocity at the edge of the boundary layer, θ its momentum thickness, L a reference length and U a reference velocity. The Reynolds number, Re_L, is based on L and U. Compare your result with the Blasius solution,

$$\frac{\theta(x)}{L} = \frac{0.664}{Re_L^{1/2}}\left(\frac{x}{L}\right)^{1/2}, \tag{11}$$

for $Re_L = 2500$ and $0 \le x/L \le 1$.

Step (i): write a Matlab function

```
f = ueintbit(xa,ua,xb,ub)
```

which will calculate the contribution to the integral in (10) between $x'/L = x_a$ (where $u_e/U = u_a$) and $x'/L = x_b$ (where $u_e/U = u_b$). Note that, if we assume a linear variation in u_e, then

$$\int_{x_a}^{x_b}\left[\frac{u_e(x')}{U}\right]^5 d\left(\frac{x'}{L}\right) = \left[\bar{u}^5 + \frac{5}{6}\bar{u}^3(\Delta u)^2 + \frac{1}{16}\bar{u}(\Delta u)^4\right]\Delta x$$

where

$$\bar{u} = \frac{u_a + u_b}{2},$$

$$\Delta u = u_b - u_a,$$

$$\Delta x = x_b - x_a.$$

Test your routine against a hand calculation of a linearly varying function between two arbitrary limits.

Step (ii): start your script by setting the Reynolds number and creating vectors of x/L and u_e/U. Use a spacing of 0.01 in x/L (`x = linspace(0,1,101)` or `x = (0:.01:1)` will do this for you), and note that, for the zero-pressure-gradient boundary layer, $u_e = U$ at all locations. (N.B. As you may have gathered from the presentation so far, it is best to write your code in terms of dimensionless variables. However, for ease of typing/reading, it is entirely acceptable to give them the same names as their dimensional counterparts, e.g. `ue` for u_e/U.)

Step (iii): Initialise the integral in (10) as zero, and then loop over (x/L) locations incrementing it accordingly. At each stage, extract (θ/L). Aim to keep your code general—if you make sure it's applicable for varying u_e/U, life will be easier in the next exercise.

Step (iv): plot your results for (θ/L) against (x/L). Also include the Blasius solution, (11), on your graph. (Remember `hold on` allows you to plot different sets of data at once, and type `help plot` to see how to use different line types and colours. You can add a key to distinguish them with the `legend` command.)

Assessment Materials: (a) listing of `ueintbit.m`,
(b) listing of script,
(c) momentum thickness plot.

Exercise 2: Constant-velocity-gradient, laminar boundary layer

Statement of exercise: Write a more general version of your Exercise 1 script, to allow linear variations in the free-stream velocity. Also include a test for transition. Tabulate transition results for velocity gradients $d(u_e/U)/d(x/L) =$ -0.2, 0 and 0.2. In each case, give results for $Re_L = 1 \times 10^6$, 10×10^6 and 100×10^6.

Step (i): edit (a copy of) your Exercise 1 script to allow you to set the velocity gradient, and thus create the array containing the linearly varying values of u_e/U. Take $u_e = U$ at $x = 0$.

Step (ii): loop over (x/L) locations calculating the (dimensionless) momentum thickness. This section of code should be directly transferable from your Exercise 1 script.

Step (iii): add a test for transition. Eppler and Somers [1] propose the empirical criterion

$$\log(Re_\theta) \geq 18.4 H_E - 21.74 \tag{12}$$

where

$$Re_\theta = \frac{u_e(x)\theta(x)}{\nu} = Re_L \left(\frac{u_e}{U}\right)\left(\frac{\theta}{L}\right)$$

is the local Reynolds number based on momentum thickness, and H_E the energy shape factor. The latter is defined in a similar way to the shape factor you have already met $(H = \delta^* / \theta)$, but in terms of the *energy thickness*, δ_E, i.e.

$$H_E = \frac{\delta_E}{\theta},$$

where

$$\delta_E = \int_0^\infty \left[1 - \left(\frac{u}{u_e}\right)^2\right] \frac{u}{u_e} dy. \tag{13}$$

Two utility routines are provided on the project Moodle site to help you code the transition check: `He = laminar_He(H)`, which inverts the relation between H and He given by Eppler and Somers [1], and `H = thwaites_lookup(m)`, which provides Thwaites' tabulated relation between the shape factor and its 'gradient parameter',

$$m = -\frac{\theta^2}{\nu}\frac{du_e}{dx} = -Re_L\left(\frac{\theta}{L}\right)^2\frac{d(u_e/U)}{d(x/L)} \ . \qquad (14)$$

This variable can be evaluated from quantities that you have already either defined or calculated.

Once you have calculated Re_θ and H_E (in the variables `Rethet` and `He`, say), you can test whether (12) is satisfied at the i-th x location, `x(i)`, with the following code:

```
if log(Rethet) >= 18.4*He - 21.74
  laminar = false;
  disp([x(i) Rethet/1000])
end
```

which also displays the location and Re_θ at which transition has occurred. The use of the variable `laminar` is explained in the following step.

Step (iv): modify the loop structure. This is necessary because the calculation cannot (yet) proceed beyond transition. You can implement this constraint by using a `while` loop in conjunction with a flag to indicate the boundary layer state, as follows:

```
n = 101;        % defines number of panels
laminar = true;   % initializes boundary layer state flag
…… other pre-loop code ……
i = 1;
while laminar && i < n
  i = i + 1;
  …… code to calculate momentum thickness ……
     …… transition check ……
end
```

Note that the comments included here after the % symbol follow the Matlab convention. You can also give comments entire lines to themselves if you wish, using the same symbol.

Assessment materials: (a) listing of script,
(b) tabulated transition locations and corresponding Re_θ values.

Exercise 3: Laminar boundary layer with transition or separation

Statement of exercise: Incorporate a separation check into your laminar boundary layer calculation. Record the separation location when $Re_L = 10^5$ and the velocity gradient is $d(u_e/U)/d(x/L) = -0.5$. Also, try $Re_L = 10^4$ and $Re_L = 10^3$. Finally, find the value of Re_L (to two significant figures) at which transition occurs before separation.

Step (i): edit (a copy of) your previous script to define two variables `int` and `ils`, for storing the location of either natural transition or laminar separation. They should be initialized to zero before entering the calculation loop.

Step (ii): extend your transition test to include a subsequent check for separation (via the `elseif` branch specifier). The condition for separation follows from Thwaites' method as

$$m \geq 0.09,$$

where m is as defined in (14). When either transition or separation is found, set `laminar = 0` (as previously), but also store the location index in `int` or `ils`, as appropriate.

Step (iii): after the loop, display the calculation result. For example, the following code does this (if necessary) for natural transition:

```
if int ~= 0
  disp(['Natural transition at ' num2str(x(int)) ...
                          ' with Rethet ' num2str(Rethet)])
end
```

(This may seem a clumsier way of handling results output than previously. However, it will be necessary subsequently.)

Assessment materials: (a) listing of script,

(b) separation locations for $Re_L = 10^5, 10^4$ and 10^3,

(c) value of Re_L at which transition supplants separation.

Exercise 4: Zero-pressure-gradient, turbulent boundary layer

Statement of exercise: Implement a numerical solution of the momentum and kinetic energy integral equations for a turbulent boundary layer with uniform free-stream velocity. Compare your calculated momentum thickness with 1/7th– and 1/9th–power–law estimates, for $\mathrm{Re}_L = 10^7$ and $0 \le x/L \le 1$.

Step (i): code the governing equations. Matlab provides a differential equation solver, `ode45`, for problems of the form

$$\frac{d\mathbf{y}}{dx} = f(x, \mathbf{y}) \tag{15}$$

where **y** is a vector of the (unknown) dependent variables and x is the independent variable. The function f must be provided by the user.

You are already familiar with the momentum integral equation:

$$\frac{d\theta}{dx} + \frac{H+2}{u_e}\frac{du_e}{dx}\theta = \frac{c_f}{2}. \tag{16}$$

The kinetic energy integral equation is derived similarly (see, for example, [2]), giving

$$\frac{d\delta_E}{dx} + \frac{3}{u_e}\frac{du_e}{dx}\delta_E = c_{diss},$$

where δ_E is the kinetic energy thickness defined in (13) and c_{diss} is a dissipation coefficient arising from the shear stresses in the boundary layer. The vector of unknowns in (15) thus consists of θ and δ_E:

$$\mathbf{y} = \begin{bmatrix} \theta \\ \delta_E \end{bmatrix}$$

and the function is given by

$$f(x, \mathbf{y}) = \begin{bmatrix} \dfrac{c_f}{2} - \dfrac{H+2}{u_e}\dfrac{du_e}{dx}y_1 \\ c_{diss} - \dfrac{3}{u_e}\dfrac{du_e}{dx}y_2 \end{bmatrix}.$$

In our implementation, we shall represent **y** by the variable `thick`, and $f(x,\mathbf{y})$ by the function `thickdash`. You thus need to code the Matlab routine

```
dthickdx = thickdash(xmx0,thick)
```

where `xmx0` represents the x variable.

In thinking about this, you will quickly come across two issues. The first is the evaluation of c_f, c_{diss} and H, which are not currently specified. Eppler and Somers [1] address this by proposing the following empirical relations:

$$H = \frac{11H_E + 15}{48H_E - 59}, \qquad H_E \geq 1.46, \tag{17a}$$

$$H = 2.803, \qquad H_E < 1.46, \tag{17b}$$

$$c_f = 0.091448[(H-1)Re_\theta]^{-0.232} e^{-1.26H}, \tag{18}$$

$$c_{diss} = 0.010025[(H-1)Re_\theta]^{-1/6}, \tag{19}$$

where H_E is the energy shape factor introduced previously (and is thus known from θ and δ_E).

The second issue is that the arguments for `thickdash` do not contain all the information needed to evaluate it (their form is constrained by `ode45`'s expectations). You will thus need to pass some global variables that allow `thickdash` to calculate u_e, du_e/dx and Re_θ. A suitable set is the free-stream velocity at the beginning of the integration range, the velocity gradient and the overall Reynolds number, Re_L. If these are called `ue0`, `duedx` and `Re`, then the statement

```
global Re ue0 duedx
```

at the beginning of `thickdash` will give you access to them.

Step (ii): begin your script by defining the global variables with the same statement as you used in `thickdash`. Matlab will complain about this if the variables already exist in your workspace, so if you haven't yet got into the habit of starting your script with `clear`, you may do so now.

Step (iii): set the values of `Re`, `ue0` and `duedx`.

Step (iv): define the initial values of θ and δ_E. This needs to be at a non-zero value of (*x/L*). The following code is suitable:

```
x0 = 0.01;
thick0(1) = 0.037*x0*(Re*x0)^(-1/5);
thick0(2) = 1.80*thick0(1);
```

Step (v): solve the differential equation with the following call to `ode45`.

```
[delx thickhist] = ode45(@thickdash,[0 0.99],thick0);
```

Here the input arguments are the name of the function defining the differential equation, the range of the independent variable (note that the latter is $x - x_0$, hence the name given to it in `thickdash`), and the initial values of the vector `thick`. The outputs consist of a vector of *x* locations, in `delx`, and the corresponding 'history' of `thick`, in `thickhist`. The latter is a matrix; its first column (which can be referred to as `thickhist(:,1)`) contains θ and its second δ_E.

Step (vi): calculate the 1/7th and 1/9th power law estimates for the momentum thickness, respectively

$$\frac{\theta_7}{L} = 0.037\left(\frac{x}{L}\right)\left[Re_L\,\frac{x}{L}\right]^{-1/5}$$

and

$$\frac{\theta_9}{L} = 0.023\left(\frac{x}{L}\right)\left[Re_L\,\frac{x}{L}\right]^{-1/6}.$$

Note that a variable containing suitable values of *x/L* can be obtained from the output of `ode45` as follows: `x = x0 + delx`.

Step (vii): plot the three sets of results for θ/*L* on a single graph.

Assessment materials: (a) listing of `thickdash.m`,
(b) listing of script,
(c) results graph.

Exercise 5: Constant-velocity-gradient, turbulent boundary layer

Statement of exercise: Write a script to calculate the evolution of a turbulent boundary layer with a linearly varying free-stream velocity, starting from $u_e = U$ at $x/L = 0.01$. Tabulate any separation locations before $x/L = 1$ for: (a) $d(u_e/U)/d(x/L) = -0.3, -0.6, -0.9$ at $Re_L = 10^7$; (b) $d(u_e/U)/d(x/L) = -0.6$ at $Re_L = 10^6, 10^7, 10^8$. Also plot θ/L and δ_E/L against x/L for the $d(u_e/U)/d(x/L) = -0.6$, $Re_L = 10^7$ case, over the range $0 \le x/L \le 1$.

Step (i): copy your script for Exercise 4 and check that your treatment of the free-stream velocity is sufficiently general for this exercise.

Step (ii): Eppler and Somers [1] quote $H_E = 1.46$ as a threshold value, below which a turbulent boundary layer will be separated. For each case, plot H_E against x/L to determine the separation locations.

Step (iii): plot the required θ/L and δ_E/L curves together on one graph.

Assessment materials: (a) listing of script,
(b) tabulated separation locations,
(c) graph.

Exercise 6: Combined laminar and turbulent boundary layer evolution

Statement of exercise: Draw up a flowchart and write a script to calculate boundary layer evolution from an initially laminar condition, through either transition or laminar separation/turbulent reattachment, to (possible) turbulent separation, over the range $0 \le x/L \le 1$. Produce the following plots:

(a) θ/L vs x/L, for $Re_L = 10^6$ and 10^7, with zero pressure gradient;
(b) H_E vs x/L for $Re_L = 10^6$ and 10^7, with zero pressure gradient;
(c) θ/L vs x/L, for $Re_L = 10^4, 10^5$ and 10^6, with $d(u_e/U)/d(x/L) = -0.25$;
(d) H_E vs x/L for $Re_L = 10^4, 10^5$ and 10^6, with $d(u_e/U)/d(x/L) = -0.25$.

On each plot, indicate locations where the boundary layer state changes. Also find, for $Re_L = 10^5$, the velocity gradient that corresponds to turbulent separation at $x/L = 1$. (Two figure accuracy is sufficient.)

Step (i): start from a copy of your fully developed laminar code (Exercise 3). Since you are asked to plot H_E, you will need to assign an arbitrary value at your first point ($x/L = 0$). Use $H_E = 1.57258$ (which corresponds to a Blasius boundary layer). Also initialize two more location recorders, `itr` (turbulent reattachment) and `its` (turbulent separation) to zero.

Step (ii): your laminar loop requires very little editing; it is only necessary to make sure that initial values are provided for the turbulent calculation. The governing equations for θ and δ_E imply that both must be continuous; equivalently we can say that θ and H_E are continuous. Both are already calculated in the laminar loop, so, in principle, no alterations are required. However, if laminar separation is detected, it is worth setting H_E explicitly to its laminar separation value, 1.51509.

Step (iii): immediately after the laminar loop, calculate δ_E ($= H_E\theta$). Note that you do not need to store your δ_E values in an array.

Step (iv): set up the `while` loop structure for your turbulent boundary layer calculation. This should proceed as long as `its` is equal to zero and your position index is less than its maximum value.

Step (v): call `ode45` as previously, but panel by panel instead of over the entire x range, i.e.

```
[delx thickhist] = ode45(@thickdash,[0,x(i)-x(i-1)],thick0);
```

(Note that `thick0` and the global variable `ue0` now correspond to values at the beginning of the panel.) Then extract θ and δ_E at the end of the panel, and test for turbulent reattachment (if laminar separation occurred and the boundary layer hasn't already reattached) and for turbulent separation. (Eppler and Somers [1] state that the boundary layer can be considered to have reattached after laminar separation once H_E increases beyond 1.58.) When either is observed, set the appropriate location indicator to the index of the current point.

N.B. For code robustness, you should test for turbulent separation even if you have not yet flagged turbulent reattachment following laminar separation.

Step (vi): complete the calculation (if necessary) after turbulent separation by assuming that the shape factor remains unchanged from its separation value. (This is an arbitrary decision, taken so that we can still obtain a complete set of results even if turbulent separation is predicted. It is necessary because Eppler and Somers' empirical closure relations, equations (17)-(19), are now no longer applicable.) Since the boundary layer is separated, $c_f = 0$, so the momentum integral equation, (16), can be solved to give

$$\left(\frac{\theta_b}{\theta_a}\right) = \left(\frac{u_{ea}}{u_{eb}}\right)^{(H+2)},$$

where the subscripts 'a' and 'b' indicate two x locations. Use this expression to complete the array of θ values.

Step (vii): output your boundary layer diagnostics, `int`, `ils`, `itr` and `its`, in the same way as you did the first two previously.

Step (viii): generate the required results.

Assessment materials: (a) flowchart
(b) listing of script
(c) four plots, as specified above
(d) critical velocity gradient, as specified above.

Reporting

Your second interim report is due at **4pm on Wednesday, 28th May**. It should take the same form as that for Week 1; i.e. the assessment materials produced by you and your partner for the Week 2 exercises, including any auxiliary functions you might have written. The submission protocol is unchanged.

Additional work for Week 2

Have you done any background reading yet?

Weeks 3 and 4: Wing Section Design

Introduction

In the final two weeks of the project, you will complete your 2D aerofoil analysis code and use it to design your own section. The latter activity is much more open-ended than the previous work, and you are encouraged to use your initiative to find useful information. A list of recommended reading is given with the references. You should also ensure that any bugs in your Weeks 1 and 2 codes have been corrected before you proceed further.

Code Completion I: Resolution study

The aim of this exercise is to assess the discretisation level required for satisfactory accuracy in your panel method. We also need to reconsider the Kutta condition. The script `resstd.m` is provided for you to use. It calls two new utility routines (also provided):

- `vdvfoil` generates a van de Vooren aerofoil geometry (see [3] for details). This test case is useful because it is obtained from a conformal mapping of a cylinder, and thus has an exact analytical solution for the surface pressures;
- `make_upanels` takes an input geometry with arbitrary point spacing and generates a set of uniform size surface panels.

All three are included in the 'week3code' zip file, which you can download from the project Moodle site.

Step (i): set the aerofoil incidence to zero and run the script. Compare the resulting computed pressure distributions with the exact solution at the trailing edge. You will observe a small wiggle in the final three points which does not disappear as the resolution is improved. This is a numerical issue, caused by the unpleasant form of the surface velocity near the trailing-edge stagnation point.

Note too that the presence of the stagnation point in the theoretical distribution implies an extreme adverse pressure gradient on the final panel. This will play havoc with your boundary layer calculation; we also know that it is not present in experimental measurements, which instead show trailing-edge pressures close to the free-stream level (c.f. Figure 5; compare the experimental data points against the 'Usual theory' curve). The reason for this,

as sketched in Figure 6, is that the actual stagnation point occurs within the boundary layer, and is inherently associated to viscous effects. Imposing it on the external, inviscid solution is not realistic.

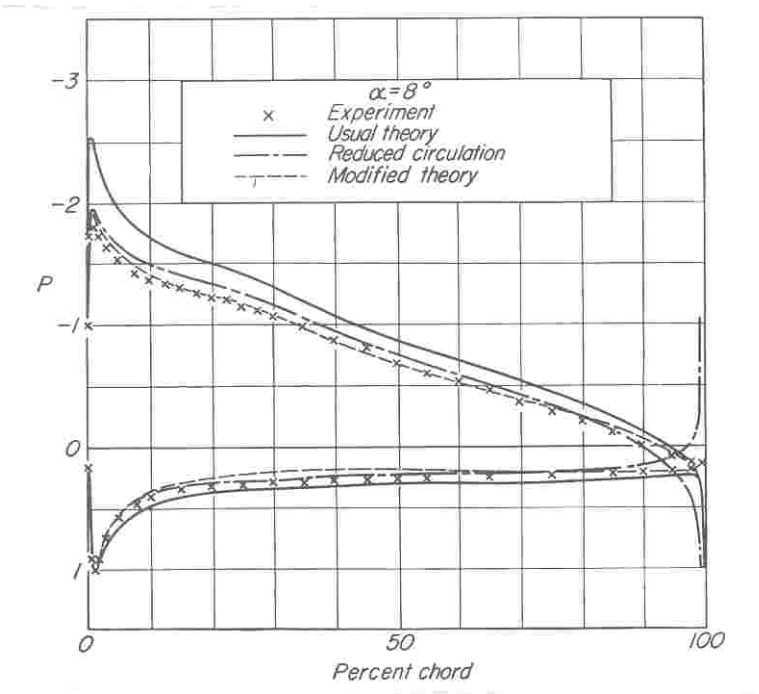

Figure 5. Calculated and measured surface pressure coefficients on a NACA 4412 aerofoil (source [4], Reynolds number unknown).

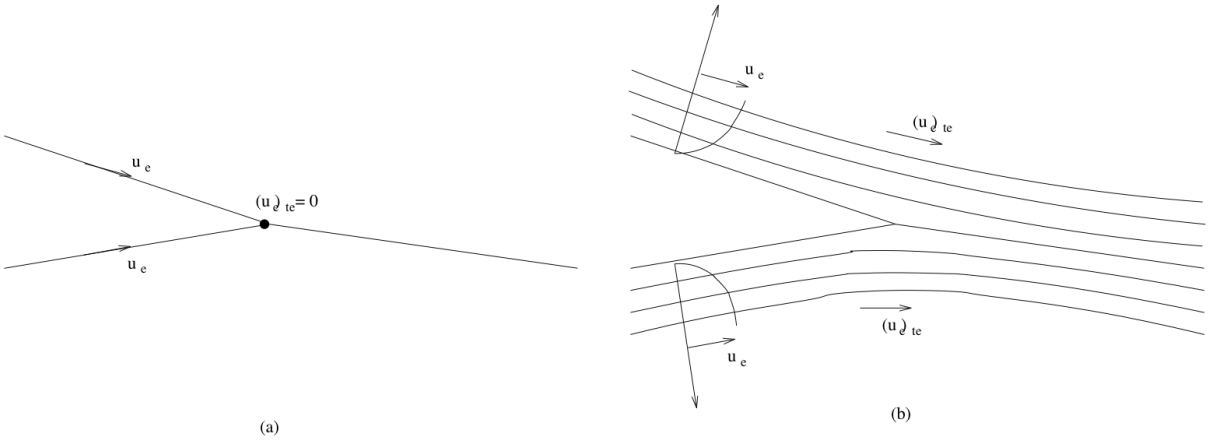

Figure 6. The effect of finite boundary layer thickness on the trailing-edge Kutta condition. (a) Potential flow solution, with upper and lower streamlines meeting at the trailing edge, which must therefore be a stagnation point. (b) Flow with thin boundary layers. The trailing edge is still a stagnation point, but the velocity in the potential flow region (which determines the trailing-edge pressure) can now be non-zero.

A more realistic approach is to impose at the trailing edge a velocity intermediate to those expected from the upper and lower surface solutions. The corresponding values of the external velocities at the trailing edge, $(u_{eu})_{te}$ and $(u_{el})_{te}$, can be obtained from linear extrapolation, as indicated in figure 7.

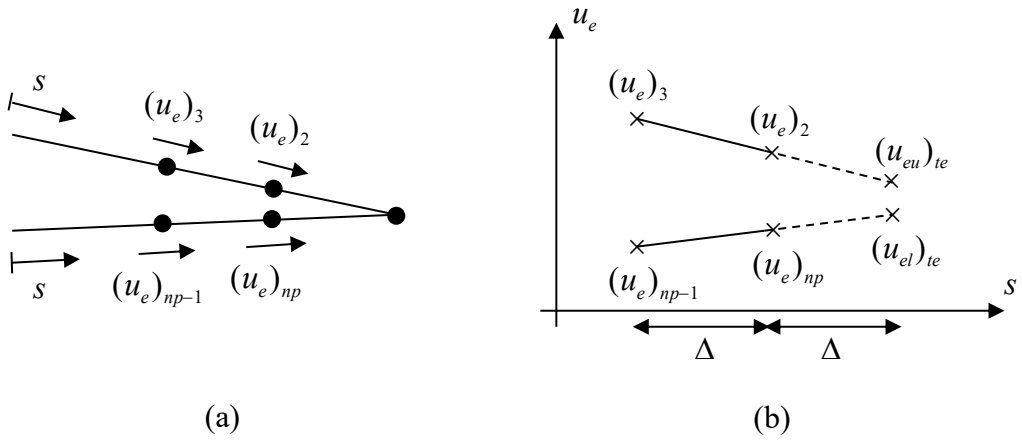

Figure 7. Modified trailing-edge Kutta condition. (a) The trailing-edge region. (b) Extrapolation of upstream velocities for trailing-edge estimates.

Our best estimate of the trailing-edge velocity is their average:

$$(u_e)_{te} = \frac{(u_{eu})_{te} + (u_{el})_{te}}{2}.$$

To work out the new 'Kutta condition' equations, you need first to express this formula in terms of the vortex sheet strengths at the relevant nodes. Once you have done this, change the associated matrix rows in `build_lhs`, and then run `resstd` again to observe the effect.

Step (ii): inspect your results around the suction peak. Then set the aerofoil incidence to $\pi/12$ (15°) and again check the suction peak fidelity.

Step (iii): edit `vdvfoil` so that it generates a thick aerofoil, instead of the thin one you have been using so far. (This involves commenting out the lines specifying the 'thin' parameter values, and uncommenting those specifying their 'thick' counterparts.) Run `resstd` again and once more observe how the solution fidelity varies with number of panels. You should now have a good idea of the resolution that will be necessary for your aerofoil designs, bearing in mind the need for numerical efficiency as well as accuracy.

Code Completion II: Boundary layer subroutine

The boundary layer calculation will be called from the main program as a subroutine;

```
function [int ils itr its delstar theta] = bl_solv(x,cp)
```

This should be based on your script for Exercise 6 of Week 2. The input information consists of surface distance from the leading edge stagnation point (in the vector `x`) and corresponding c_p values. The effective geometry is shown in Figure 8.

Step (i): **print out your Exercise 6 script for editing**. Converting it to a subroutine requires careful thought at some points, and trying to edit the electronic version directly is likely to lead to bugs. A bit of extra effort now will save you a lot of wasted time later!

Step (ii): work out the number of panels from the length of `x`; also remove your statement defining the uniform velocity gradient.

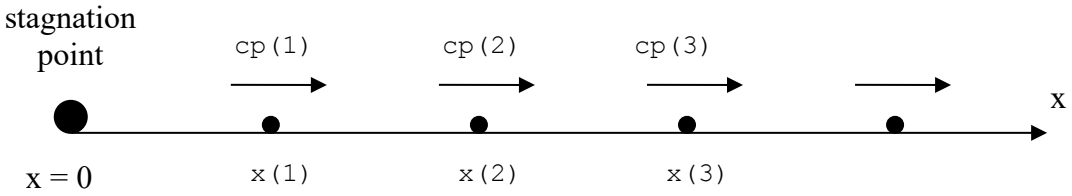

Figure 8. Effective flat plate geometry for the aerofoil boundary layer calculation.

Step (iii): edit your laminar loop to: include a calculation of the current u_e from the current c_p; estimate the panel velocity gradient; and evaluate the value of δ^* for entry in the `delstar` array. Note that $u_e/U = \sqrt{1-c_p}$.

Step (iv): revisit your initialization code. Previously $i = 1$ was the beginning of the boundary layer, but now (Figure 8) it is the next point. This means that your initialization needs to be an edited version of the code in your loop.

Step (v): edit your turbulent loop to: include a calculation of the current u_e; estimate the panel velocity gradient; and evaluate the value of δ^*.

Step (vi): edit your turbulent separated loop to include a calculation of the current u_e and evaluate the value of δ^*.

Step (vii): remove the output stages of the original script.

Step (viii): test your new routine by recreating some of your zero-pressure-gradient boundary layer results from Week 2. The script `test_bl_solv` is provided for this purpose. NB The boundary-layer calculation also requires the Reynolds number. Check how `test_bl_solv` makes it available, and ensure that your subroutine will receive it correctly.

Code Completion III: Forces on the aerofoil

The last step is to write a function to calculate the lift and drag forces on the aerofoil:

```
function [cl cd] = forces(circ,cp,delstarl,thetal,delstaru,thetau)
```

in which:

`circ` is the dimensionless circulation, Γ/Uc;

`cp` is the array of pressure coefficients, indexed in the same way as the surface points;

`delstarl` is the lower surface displacement thickness array;

`thetal` is the lower surface momentum thickness array;

`delstaru` is the upper surface displacement thickness array;

`thetau` is the upper surface momentum thickness array.

Step (i): the lift is related to the circulation by the Kutta-Joukowski theorem. In dimensionless form, this is

$$C_L = -2\frac{\Gamma}{Uc},$$

where c is the aerofoil chord. This replaces L as our reference length.

Step (ii): a control volume analysis (see, for example, [2]) shows that the drag is proportional to the momentum thickness of the wake 'at infinity' (i.e. where the external velocity has recovered to its free-stream value). In dimensionless form:

$$C_D = 2\frac{\theta_\infty}{c}$$

The momentum thickness at infinity can be estimated from the momentum thickness at the trailing edge, θ_{te}, via the Squire-Young relation [2]:

$$\theta_\infty = \theta_{te}\left[\frac{(u_e)_{te}}{U}\right]^{\frac{H_{te}+5}{2}},$$

where H_{te} is the shape factor at the trailing edge. The trailing edge displacement and momentum thicknesses can be found by summing the upper and lower surface boundary layer contributions at this point. The trailing edge velocity is (implicitly) specified by the pressure coefficient there.

Validation

You should now have available to you all the subroutines required by the main program, which is provided for you in `foil.m`. **Have a look at this code, and make sure you understand what it does.** You will need to create subfolders 'Parfiles' and 'Geometry' inside your working directory, and copy into them the files 'tryout.txt' and 'foil.surf', respectively, from Moodle. To check that everything is functioning, type `foil` at the Matlab prompt and then `tryout` in response to `Enter case reference:`. This will run the code with the parameters specified in tryout.txt. Note that the format of tryout.txt should not be modified when you edit this file for subsequent cases (check how it is read in to see why). Finally, observe the data output statements in `foil.m`, and check that the files you would expect have been produced.

Once the code runs smoothly, test its predictions against experimental results for two NACA sections of 'typical' 12% thickness-to-chord ratio: the uncambered 0012 and the cambered 4412. A utility routine to generate NACA four-digit sections is provided in `naca4.m`. As an example

```
[xk yk] = naca4('0012')
```

will generate the surface coordinates for the NACA0012. It will also save the coordinates in the file 'Geometry/naca0012.surf', with format appropriate for reading by `foil`. Copies of experimental results graphs for the two sections are available in Moodle; a suitable Reynolds number for the comparison is 3×10^6.

Try not to be content with simply observing the extent of differences between experiment and calculation; this is also a good opportunity to gain further insight into both aerofoil flows and your program's characteristics. The following questions are intended to help you do so.

$C_L - \alpha$ *curves*

- What do these curves look like over typical aerofoil incidence angles?
- How do they depend on Reynolds number?
- What is the influence of camber?
- What happens at very high incidence angles?
- How do the curves compare with experimental results?
- Can you explain the differences?

$C_D - \alpha$ *curves*

- What do these curves look like?
- How do they depend on Reynolds number?
- What is the influence of camber?
- Can you link variations in C_D with the information available from your boundary-layer subroutine output?

$C_D - C_L$ *curves*

- How do these curves compare with experimental results?
- What are the sources of error?
- Does your code output give you any warning of cases that are likely to be inaccurate?
- In what incidence region are high lift-to-drag ratios achieved?
- How accurate are your predictions in this region?
- Challenge: should you assess your drag predictions on the basis of a $C_D - \alpha$ comparison instead of $C_D - C_L$?

Design

The aim of the design process is to produce sections with good performance. The Appendix shows that the relevant measure is the lift-to-drag ratio. You should thus seek to optimize this parameter. As a starting point, do this for two different Reynolds numbers, 0.5×10^6 and 20×10^6, to obtain a 'low-speed' and a 'high-speed' design.

It is tempting to begin with a systematic variation of camber-line and thickness functions. This is *not* a good idea, for two reasons: it has already been done by NACA (exhaustively, see [4]), and it turns out to be a relatively inefficient approach. It is better to start with a rough geometry and use your understanding of the factors governing aerofoil performance to make incremental modifications to the surface coordinates. A Wing Analysis Section Generator — wasg.m — is provided with the Week 3 code files for this purpose. Instructions for its use are given in Appendix 2.

Once you have your two optimised section designs, test the low-speed one at Reynolds number 20×10^6, and the high-speed at 0.5×10^6. Having done so, you will have sufficient results to form the basis of your final report. You may, however, have the time and inclination to go further. If so, here are some possible avenues of investigation: alternatives to your original design approach; optimising for high lift-to-drag over a range of incidences; section design for good performance at both Reynolds numbers. Equally, feel free to pursue other ideas that you may have (though it may be worth discussing these with the project demonstrators first).

Finally, a hint: you will find it extremely difficult to link section geometry with lift and drag characteristics directly. The key intermediary is the pressure distribution; try to understand how it is affected by geometry changes, and how it in turn affects lift and drag.

Reporting

There is no limit (minimum or maximum) on the size of your final report. However, you should aim to keep it concise, and to use figures/diagrams as much as possible to get your points across. Do not attempt to reiterate basic aerofoil theory that can be easily referenced in textbooks; credit will instead be given for the level of understanding of your results, in the context of this theory, that you exhibit. Equally, this aspect of your report will be treated as more important than the absolute lift-to-drag ratios achieved by your design.

Your report should address the following subjects:

- the validity and accuracy of the numerical model;
- the design process by which you reached your optimum aerofoil section(s);
- the optimum sections and their aerodynamic characteristics;
- the physical processes which determine the aerodynamic characteristics of the optimum (and non-optimum) sections;
- the reliability of your results.

The report is due by **4pm** on **Thursday, 12th June** and should be submitted electronically via **Moodle**. Results in partners' reports can be identical, but the reports themselves must be individual work.

Background Reading and References

References (with CUED shelfmarks where applicable)

[1] Eppler & Somers, 'A computer program for the design and analysis of low-speed airfoils', NASA TM 80210.

[2] Young, A D, 'Boundary layers', BSP Professional Books, TA374.

[3] Katz & Plotkin 'Low speed aerodynamics', McGraw Hill, TD128.

[4] Abbott & von Doenhoff 'Theory of Wing Sections', Dover, TD61.

[5] Barnard & Philpott, 'Aircraft Flight', Longman, TQ88.

[6] Simons, 'Model aircraft aerodynamics', Argus Books, TD142.

[7] McGhee et al, 'NASA low- and medium-speed airfoil development', in NASA CP 2046, 'Advanced technology airfoil research Vol. II'.

[8] Hoerner, 'Fluid dynamic drag', Hoerner Fluid Dynamics, TA115.

[9] Hoerner, 'Fluid dynamic lift', Hoerner Fluid Dynamics, TD122.

The origin of lift

[5] Chapter 1

The source of drag; separation and stall

[5] Chapter 3

Boundary layers

[2]

Aerofoil section characteristics and design

[4] Chapter 7

[6] Chapters 7–9

[7]

[8] Chapters II & VI

[9] Chapter II

Appendix 1: Background

The significance of the lift-to-drag-ratio

Consider an aircraft in cruise flight, so that its lift, L, equals its weight, W, and its drag, D, equals its thrust, T. The aircraft weight changes due to fuel consumption, according to the equation

$$\frac{dW}{dt} = -g\text{sfc}T = -g\text{sfc}D = -g\text{sfc}\frac{L}{(L/D)} = -g\text{sfc}\frac{W}{(L/D)},$$

where sfc is the *specific fuel consumption*, i.e. the mass flow rate of fuel per unit of thrust. Now $dW/dt = dW/dx \times dx/dt$, so we have

$$U\frac{dW}{dx} = -g\text{sfc}\frac{W}{L/D},$$

where U is the flight speed. If U is fixed, and we neglect variations in sfc and L/D (an approximation, but a reasonable one in practice for constant speed flight), then we can integrate to obtain

$$\text{Range} = x_{end} - x_{start} = \frac{U}{g}\frac{L/D}{\text{sfc}}\log\left(\frac{W_{start}}{W_{end}}\right).$$

This is the *Breguet range equation*.

Although the assumptions behind this equation will not be satisfied exactly in practice, it neatly highlights the ways in which aerodynamics, power-plant and aircraft structure influence the aircraft efficiency, through L/D, sfc and W_{start}/W_{end} respectively. The wing designer thus seeks to maximise L/D, the lift-to-drag ratio.

Figure A1, taken from [4], shows a typical example of the variation in L/D of a 2-D wing section with angle of attack, α. As α increases from the zero-lift angle, L/D rises to a maximum, and then falls again. This fall is mainly associated with a significant increase in drag, but is exacerbated at higher angles of attack by a reduction in lift. The maximum value attained is about 32, but much higher values can be reached – many of the sections in [4] have maximum L/Ds of over 100.

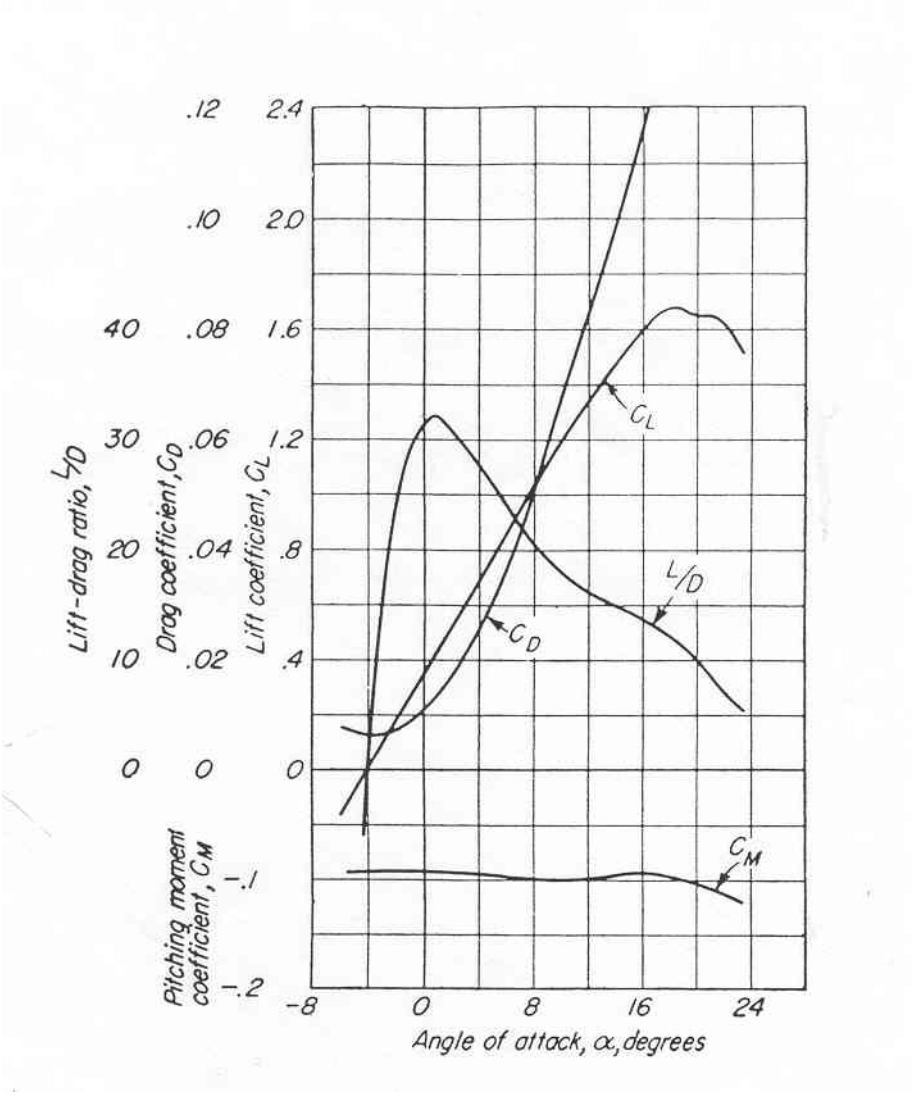

Figure A1. Typical aerofoil section force and moment variations with incidence [4].

For real aircraft, fuselage drag and three-dimensional effects on the wing reduce the maximum *L/D*; Figure A2 shows that a Boeing 747 has a maximum *L/D* of about 18. (Note, in passing, how compressibility effects reduce *L/D*, and hence limit the maximum practical cruise Mach number.) However, the 2-D section drag still plays a significant part in determining this maximum, and improvements in the 2-D section efficiency will feed through into improvements in aircraft efficiency.

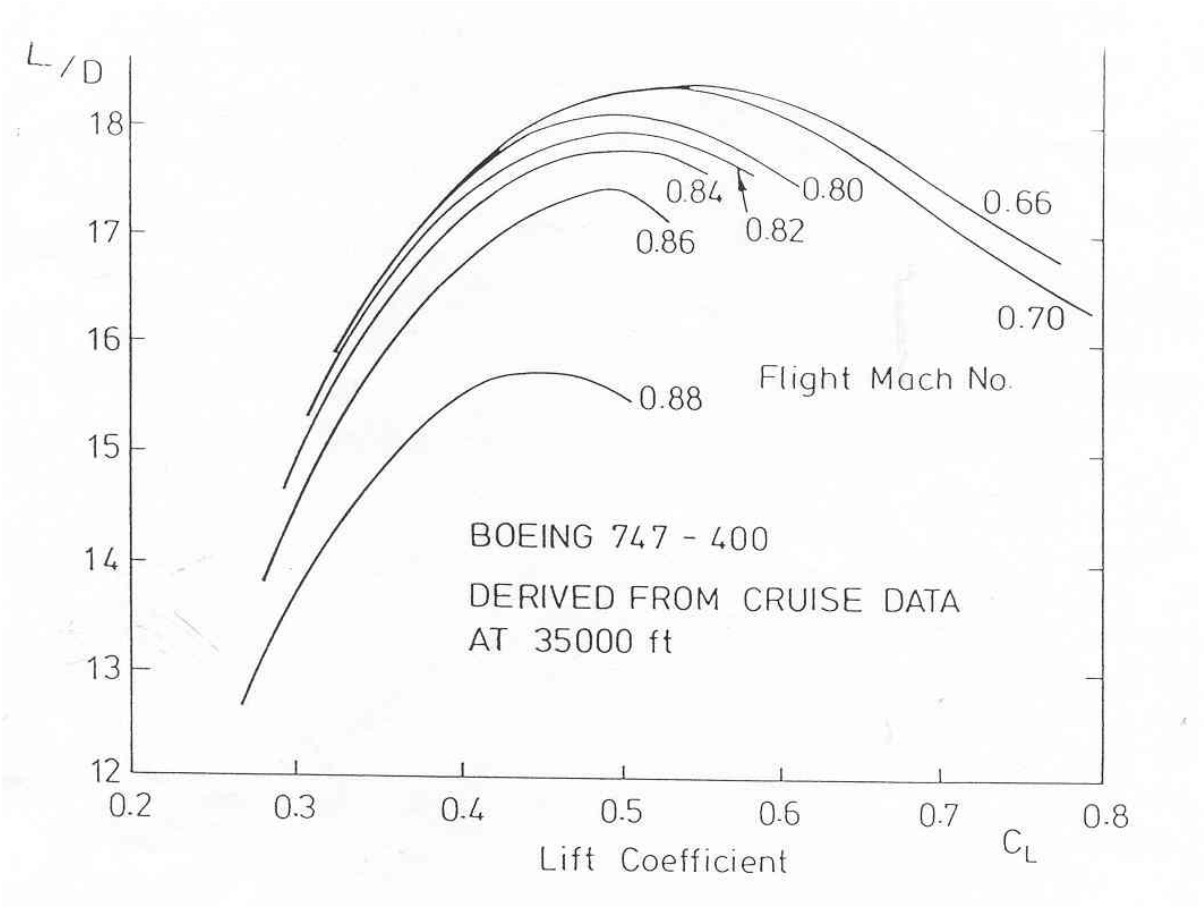

Figure A2. Lift-to-drag ratios for the Boeing 747 (Cumpsty, N.A., 'Jet Propulsion', CUP, 1997).

The wing lift

The wing lift is defined as the component of wing force perpendicular to the free-stream velocity, and is usually expressed in terms of a dimensionless lift coefficient;

$$C_L = \frac{L}{\frac{1}{2}\rho U^2 S},$$

where ρ is the air density, U the free-stream velocity and S the wing area. Lift arises because the air flows round the upper surface at a higher speed than the lower, generating a pressure difference across the wing.

A popular explanation for the higher speed on the upper surface states that the air taking this route has 'further to travel'. However, this implicitly relies on the assumption that two neighbouring particles of air which subsequently pass either side of the wing must meet up again downstream, which is not the case. The explanation is rather more subtle; a clear statement of it can be found in [5]. Suffice it to say that, of the three possible attached trailing edge flows shown in Figure A3, it is the second, where the flow leaves the trailing edge smoothly, which is observed to occur, and this flow pattern entails higher air speeds on the upper surface.

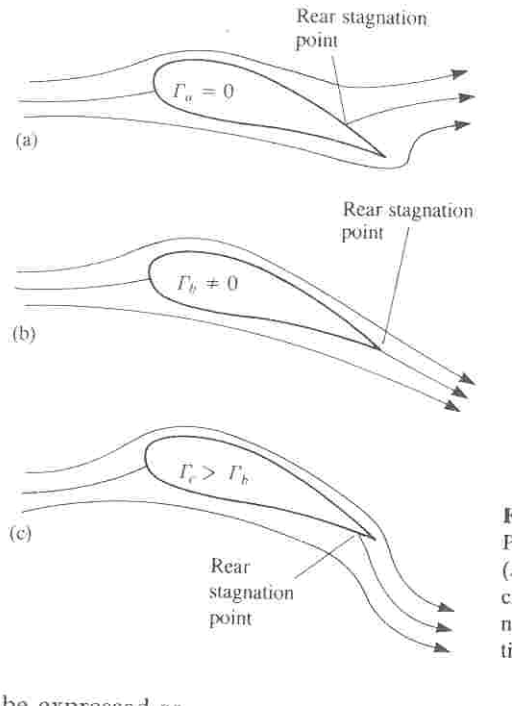

Figure A3. Theoretically admissible solutions for the potential flow around an aerofoil [3].

At high angles of attack, the flow is found to detach from the upper surface before it reaches the trailing edge. The flow detachment tends to reduce the velocities on the upper surface, and hence causes the drop in lift seen in Figure A1. This phenomenon is known as 'stall'.

The wing drag

The wing drag is defined as the component of wing force parallel to the free-stream velocity, and is characterised by the drag coefficient:

$$C_D = \frac{D}{\frac{1}{2}\rho U^2 S}.$$

On a 2-D section it arises from two sources; the shearing stresses on the aerofoil surface (*skin friction drag*) and the pressures (*form* or *pressure drag*). Both contributions are due to the presence of a thin *boundary layer*, where viscous effects are important. (The boundary layer is responsible for pressure drag in that it slightly modifies the pressure distribution on the aerofoil; in the absence of a boundary layer the pressure force on the aerofoil would have no drag component.)

The behaviour of the boundary layer is crucially dependent on the pressure distribution round the aerofoil; specifically, on the pressure gradient as the surface is traversed in the flow direction. Different pressure distributions can have markedly different effects on the boundary

layer and thus on drag – hence the importance of the aerofoil shape in determining its efficiency.

If the boundary layer reaches a state where the shear stress at the aerofoil surface is zero, then it will detach. This is the origin of stall, and it typically results in a large increase in drag as well as the loss of lift previously described. The nature of the stall is again crucially dependent on the aerofoil shape.

The design problem

The fundamental requirement of a wing, i.e. the provision of lift, inevitably leads to a pressure distribution which has adverse effects on the boundary layer at some point. It is the designer's job to minimise these effects, thereby maximising the lift-to-drag ratio. Furthermore, this should be achieved over as wide a range of operating conditions as possible.

Appendix 2: The Wing Analysis Section Generator

The `foil` program requires the section geometry at zero angle of attack as input. This is passed to it in the file '*id*.surf' in the 'Geometry' subdirectory, where *id* is the section name. The section generator allows you to define a geometry via a set of cubic spline curves linked by 'knots' and then save it in the required format. The knots can be manipulated as follows:

move: place cursor on knot, then click on mouse button 1 and drag;
add: place cursor at desired location and click on mouse button 1;
delete: place cursor on knot and click on mouse button 2.

To begin, run wasg.m and load any '*id*.surf' file from the 'Geometry' subdirectory window that comes up. (It can subsequently be summoned by pressing the 'l'-letter key.) It is possible to undo/redo knot manipulations by pressing 'u'/'r'. The present configuration can be backed up into a buffer ('b') and restored ('t') from it. To save a section press 's', then specify the destination file name – '.surf' will automatically be appended to the specified name. A zoomed-in detail window centred about the present position of the cursor will open/close when pressing 'z'. Operations can only be conducted in the zoomed-in window while it is open. For fine control of a knot's position, the arrow keys can be used to displace the selected knot in intervals of length delta ('d').

Warning! For unusual shapes, particularly those with very few knots, the displayed aerofoil section may be significantly different from that passed to the calculation; you can check the latter afterwards with `plot(xs,ys)`.