MATLAB® Coursework Part 1: Airfoil

This script has been commmented as such so that the file can be published completely

However the script will take significantly longer due to formatting published document

Contents

HOUSEKEEPING

VARIABLES:

- **screen_ratio -** Array containing ratios of CURRENT screensize to ORIGINAL screensize program was written on
- **xratio -** The airfoil 4 series desgination
- **yratio -** The airfoil 4 series desgination

```
clear
clc

% Relative screen size ratios from user to original pc (program was written on)
screen_ratio=disp_transform(10,10)./10;
xratio=screen_ratio(1);
yratio=screen_ratio(2);

% To remove orange warning signs from command window  in reference to the poorly conditioned matrices and have clearer output
warning("off")
```

CREATING AIRFOIL

User input of airfoil and freestream characteristics

VARIABLES:

- **NACA -** The airfoil 4 series desgination
- **n -** Panel number airfoil will be discretized into
- **alpha -** Angle of attack of airfoil
- **uinf -** Freestream velocity
- **Prompt -** String within input dialogue boxes
- **title_text -** String for title of input dialogue box
- **errstr -** String to display within error box
- **WARNBOX -** Handle for error box if incorrect input
- **ERRORBOX -** Handle for error box if XFOIL file not present
- **XFOIL -** Array of xfoil data from .txt file

Code has been setup so that:

ONLY IF XFOIL .TXT FILE IS IN SAME FOLDER AS SCRIPT:

script will run if NACA2412 chosen

```
% fprintf() allows us to enter a blank line in command window for clearer output
fprintf(1,'\n');

% When using dialogue boxes for input the outputs are always character strings within cell arrays or doubles

%  Following conditions used for error catching and activating while loops
NACA = {0000};

% Making sure NACA entry fits conditions
% mod(str2double(NACA{1}),1) > 0 means that input doesnt include decimal
% length(NACA{1}) ~= 4 means input is always 4 characters
% str2double(NACA{1}) <= 0 means input is always postive number
% str2double(extractAfter(NACA{1},2)) == 0 means NACA doesnt end in 00
% isnan(str2double(NACA{1})) means input is always a number
while mod(str2double(NACA{1}),1) > 0 || length(NACA{1}) ~= 4 || str2double(NACA{1}) <= 0 || str2double(extractAfter(NACA{1},2)) == 0 || isnan(str2double(NACA{1

    % Setting prompt for input dialogue box
    Prompt = 'What is the NACA 4-Series airfoil you would like to display? ';
    % Setting title for dialogue box
    title_text = 'NACA';

    % Setting dialogue box
    % [1,(length(Prompt)+50)] allows for dialogue box to fit to text
    NACA = inputdlg(Prompt,title_text,[1,(length(Prompt)+50)],{'2412'});

    % if pinput still isnt within conditions then display warning
    if mod(str2double(NACA{1}),1) > 0 || length(NACA{1}) ~= 4 || str2double(NACA{1}) <= 0 || str2double(extractAfter(NACA{1},2)) == 0 || isnan(str2double(NACA{

        % Displaying error when NACA entry isn't a positive 4 digit entry that has no decimals or ends with 00
        % Setting string for error box
        errstr = {'After clicking ok;','Please pick a known NACA 4 series airfoil',"e.g. a positive 4 digit number that doesn't end with 00"};

        % Setting title for error box
        title_text = 'NACA ERROR';

        % Forming error box
        WARNBOX = warndlg(errstr,title_text,[1,length(errstr)+50]);

        % Waiting for button press on error box
        waitfor(WARNBOX)

    end

end

% If NACA2412 chosen then panel number must be 200
if NACA{1} == num2str(2412)

    % Special case for 2412
    n = {'200'};

    % Checking if XFOIL File is in same folder - if available the script WILL READ .TXT file
    if isfile('xf-naca2412-il-1000000.txt')

        % This line of code will verify that the xfoil text file has been read in any given run of the code
        disp('The xf-naca2412-il-1000000.txt WAS READ.')
        fprintf(1, '\n');

        % Reading xfoil data as it is required to do CL_Alpha plot later
        XFOIL = table2array(readtable('xf-naca2412-il-1000000.txt'));

    else

        % Setting string for error box
        errstr = {"The Xfoil .txt file wasn't presesnt in same folder",'Please place xf-naca2412-il-1000000.txt in same folder as this script '};

        % Setting title for error box
        title_text = 'XFOIL FILE ERROR';

        % Forming error box
        ERRORBOX = errordlg(errstr,title_text,[1,length(errstr)+50]);

        % Waiting for button press on error box
        waitfor(ERRORBOX)
        clear

        % Stops scipt to wait for user to move .txt file
        return

    end

else

    % Activating while loop
    n = 0.1;

    % Putting panel number input fits conditions
    % mod(str2double(n),2) ~= 0 means that input doesnt include decimal
    % str2double(n) < 1 means input is always positive number
    % isnan(str2double(n)) means input is always a number
    while mod(str2double(n),2) ~= 0 || str2double(n) < 1 || isnan(str2double(n))

        % Setting prompt for input dialogue box
        Prompt = 'How many even number of panels would you like to display the airfoil with? ';

        % Setting title for dialogue box
        title_text = 'Panel Number';

        % Setting dialogue box
        n = inputdlg(Prompt,title_text,[1,(length(Prompt)+50)],{'200'});

        if mod(str2double(n),2) ~= 0 || str2double(n) < 1 || isnan(str2double(n))

            % Displaying error when panel number isn't a positive even number
            % Setting string for error box
            errstr = {'After clicking ok;','Please pick a positve even number for panel number','e.g. 200 panels'};

            % Setting title for error box
            title_text = 'PANEL NUMBER ERROR';

            % Forming error box
            WARNBOX = warndlg(errstr,title_text,[1,length(errstr)+50]);

            % Waiting for button press on error box
            waitfor(WARNBOX)
```

```matlab
        end

    end

end

% No conditions for alpha since angle of attack can be any number
if NACA{1} == num2str(2412)

    % Special case for 2412 with default preset values
    alpha = {'10'};
    uinf = {'15'};

else

    alpha = {'abc'};

    % Putting panel number input fits conditions
    % isnan(str2double(n)) means input is always a number
    while isnan(str2double(alpha))

        % Setting prompt for input dialogue box
        Prompt = 'At what angle of attack is the airfoil in degrees? ';

        % Setting title for dialogue box
        title_text = 'Angle of Attack';

        % Setting dialogue box
        alpha = inputdlg(Prompt,title_text,[1,(length(Prompt)+50)],{'10'});

        if isnan(str2double(alpha))

            % Displaying error when panel number isn't a positive even number
            % Setting string for error box
            errstr = {'After clicking ok;','Please pick a  number for angle of attack in degrees','e.g. 10 degrees'};

            % Setting title for error box
            title_text = 'ANGLE OF ATTACK ERROR';

            % Forming error box
            WARNBOX = warndlg(errstr,title_text,[1,length(errstr)+50]);

            % Waiting for button press on error box
            waitfor(WARNBOX)

        end

    end

    % Activating next while loop below for velocity question if NACA isn't 2412
    uinf = {'0'};

end

% Making sure velocity fits input conditions
% mod(str2double(uinf),0) == 0 makes sure velocity isnt 0
% isnan(str2double(n)) means input is always a number
% str2double(uinf) < 0 checks if negative number still allows to be plotted
while mod(str2double(uinf),0) == 0 || isnan(str2double(uinf))

    % Setting prompt for input dialogue box
    Prompt = 'What is the freestream velocity? ';

    % Setting title for dialogue box
    title_text = 'Freestream Velocity';

    % Setting dialogue box
    uinf = inputdlg(Prompt,title_text,[1,(length(Prompt)+50)],{'15'});

    % Displaying various warnings for when velocity isn't a positve number
    if mod(str2double(uinf),0) == 0 || str2double(uinf) < 0 || isnan(str2double(uinf))

        % Displaying warning when NACA entry isn't a positive number
        % Setting string for warning box
        % Default warning string assuming velocity input was 0
        errstr = {'After clicking ok;','Please pick a number for the freestream velocity','with magnitude greater than 0 for a more realistic display of airfl

        if str2double(uinf) < 0

            % Resetting string for incase velocity is 0 to allow for
            % negative airflow
            errstr = {'After clicking ok;','The flowfield display with a negative velocity','showing a backwards airflow'};

        end

        % Setting title for error box
        title_text = 'Velocity ERROR';

        % Forming error box
        WARNBOX = warndlg(errstr,title_text,[1,length(errstr)+50]);

        % Waiting for button press on error box
        waitfor(WARNBOX)

    end

end
```

```
% Reconverting input cell arrays into either strings or doubles for easier use
n = str2double(n);
alpha = str2double(alpha);
uinf = str2double(uinf);
NACA = NACA{1};
```

Discretising chosen airfoil

VARIABLES:

- **xi -** X values for panel end points including wake
- **zi -** Z values for panel end points including wake

Panelgen() attached in same zipped folder as this script

```
[xi,zi] = panelgen(NACA,n,alpha);
```

Displaying discretised airfoil

VARIABLES:

- **stream_plot -** Figure used to display aerofoil and flowfield
- **panels -** Plot to display the airfoil discretized into chosen panels
- **crit_panel** - Lowest panel number for plot to be smooth
- **line_colour -** String array storing colour and style for Chosen airfoil

```
% Multiplying relative size of figure to original pc with users pc to get desired visual
stream_plot = figure('Color',[0.9,0.9,0.9]);

% Plotting the airfoil with the chosen panel number e.g. at low panel number airfoil will not be shown as smooth
crit_panel = 20;

if n < crit_panel

    line_colour = '-k';

else

    line_colour = 'r';

end

panels = plot(xi(n/2+1:n+1),zi(n/2+1:n+1),line_colour,'LineWidth',2,'DisplayName','Discretisied Airfoil');
hold on
plot(xi(1:n/+1),zi(1:n/+1),line_colour,'LineWidth',2);

% Plotting styles so visuals are clear: font is made readable and line size is preset from handout
% Setting gca allows us to change thickness of whatever is on "current axes"
set(gca,'FontSize',15)
title(['NACA',NACA,' with ',num2str(n),' panels at ',num2str(uinf),' m/s',' and at an {\alpha} of ',num2str(alpha),char(176)],'FontSize',15)

% making blank background
grid off

% Setting axes labels
xlabel('Horizontal distance from leading edge x/c','FontSize',15)
ylabel(['Vertical distance from leading edge  ','\it\color{black}','z/c'],'FontSize',15)

% Whilst flowfield is set specifically, plot is made so equal axis like handout
ylim([-0.7,0.7])
xlim([-0.3,1.3])
```

Calculating coefficient of lift an panel strengths

VARIABLES:

- **N_Panel_MUs -** Panel strengths for each of the n number of panels
- **Cl** - Coefficient of lift calculated from the strength of the wake panel
- **norm_black -** Font type: Normal with black colour
- **bold_black -** Font type: Bold with black colour
- **bold_italic_black -** Font type: Bold & Italic with black colour
- **ital_black -** Font type: Italic with black colour
- **norm_red -** Font type: Normal with red colour
- **bold_red -** Font type: Bold with red colour
- **norm_blue -** Font type: Normal with blue colour
- **bold_blue -** Font type: Bold with black colour
- **str -** Cell string array with label for calculated Cl
- **Cl_textbox -** Handle for Estimated CL textbox

finalcalc() attached in same zipped folder as this script

```
% Using finalcalc() to find out panel strengths and cl
[N_Panel_MUs,Cl] = finalcalc(xi,zi,uinf);
disp(['The estimated coefficient of lift is ',num2str(Cl)])
fprintf(1,'\n');
```

```
% Setting variables with latex style code to take place of font types in textbox
norm_black = '\rm\color{black}';
bold_black = '\bf\color{black}';
bold_italic_black = '\bf\it\color{black}';
ital_black = '\it\color{black}';
norm_red = '\rm\color{red}';
bold_red = '\bf\color{red}';
norm_blue = '\rm\color{blue}';
bold_blue = '\bf\color{blue}';

% Checking if Xfoil data is available to compare to NACA
if str2double(NACA) == 2412 && alpha >= -17.5 && alpha <= 17.5 && any(XFOIL(:,1) == alpha)

    % XFOIL((XFOIL(:,1)==alpha),2) allows us to check any alpha input against xfoil table and get corresponding Cl
    % The following set of strings are used to display the coefficients of lift on the flowfield plot
    disp(['The actual (XFOIL) coefficient of lift is ',num2str(XFOIL((XFOIL(:,1) == alpha),2))])
    str1 = [norm_black,'The',bold_red,' estimated',bold_italic_black,' C','\rm',bold_black,'_{L}',norm_black,' is ',bold_red,num2str(Cl)];

    % The following strings are concatenated into one
    str2 = [norm_black,'The',bold_blue,' actual (XFOIL) ',bold_italic_black,'C','\rm',bold_black,'_{L}',norm_black,' is ',bold_blue];
    str2 = [str2,num2str(XFOIL((XFOIL(:,1) == alpha),2))];

    % Collecting strings for 2 columns of 1 textbox
    str = {str1,str2};
    clear str1 str2

    % Setting handle annotation for CL textbox for easy copying onto quiver
    % plot
    Cl_textbox = annotation('textbox',[0.1425,0.23,0,0],'String',str,'FitBoxToText','on','BackgroundColor',[1 1 1],'FontSize',15,'LineWidth',3);

else

    % If not 2412 then only need to annotate with estimated cl
    str = {[norm_black,'The',bold_red,' estimated',bold_italic_black,' C','\rm',bold_black,'_{L}',norm_black,' is ',bold_red,num2str(Cl)]};
    Cl_textbox = annotation('textbox',[0.1425,0.185,0,0],'String',str,'FitBoxToText','on','BackgroundColor',[1 1 1],'FontSize',15,'LineWidth',3);

end

% Adjusting size of textbox according to screensize ratio
Cl_textbox.FontSize=Cl_textbox.FontSize*xratio;
fprintf(1,'\n');

% Retaining old data (chosen N number) before clearing irrelevant data and redoing at higher n for flowfield data
[NACA_chosen,x_panel_pts,z_panel_pts,n_panel_strengths,n_cl,panel_number] = deal(NACA,xi,zi,N_Panel_MUs,Cl,n);
```

Repeat at higher N in order to generate only show flow outside of this smooth shape

VARIABLES:

- **Variables** below have been explained previously
- **order_choice -** Output for question regarding plot order for small panel numbers

```
% Clearing irrelevant data and keeping required variables
clear xi zi N_Panel_MUs NACA Cl n

% Smooth airfoil required so panel number is now 300
n = 300;
[xi,zi] = panelgen(NACA_chosen,n,alpha);

% Code to display both smooth and chosen panel number airfoil
if panel_number < crit_panel

    figure(stream_plot)
    smooth_airfoil = plot(xi(n/2+1:n+1),zi(n/2+1:n+1),'-r','LineWidth',2,'DisplayName','Airfoil','displayname','Smooth Airfoil');
    hold on
    plot(xi(1:n/+1),zi(1:n/+1),'-r','LineWidth',2);
    order_choice = questdlg('Would you like to see the airfoil prduced by your low panel number (black) on top of the smooth airfoil (red)','Airfoil Order','Y

    if all(order_choice == 'Yes')

        % Code to bring chosen panel number (black) airfoil to front of plot by
        % reordering plots on figure based on user choice
        stream_plot_CHILDREN = get(stream_plot.CurrentAxes,'Children');
        set(stream_plot.CurrentAxes,'Children',[stream_plot_CHILDREN(3:4),stream_plot_CHILDREN(1:2)])

    end

end
```

Discretising plot grid into points with individual velocities for flow

VARIABLES:

- **Variables** below have been explained previously
- **stream_res -** Number of points along one side of grid
- **mesh_x -** X values for each point on grid which has been discretised
- **mesh_z -** Z values for each point on grid which has been discretised
- **thick_perc_inc -** Percentage increase for NACA
- **NACA -** Temporary NACA to exclude points on grid for flowfield at a slightly thicker airfoil
- **in -** Boolean array matching grid size where 1s are points in the airfoil

- **on -** Boolean array matching grid size where 1s are points on the airfoil

```
% stream_res is how many points the width and height of plot have been split into each
% 100 is an arbritary value to outputs a clean flow
stream_res = 100;

% Assigning length and hieght intervals of future mesh
mesh_x = linspace(-0.2,1.2,stream_res);
mesh_z = linspace(-0.7,0.7,stream_res);

% meshgrid() allows you to make an x and z row vector and then create all the grid points defined by them
[mesh_x,mesh_z] = meshgrid(mesh_x,mesh_z);

% Inpolygon() used to check whether points on grid are in or on the airfoil *defined by the smooth airfoil (high panel number)*
clear xi zi

% If the max thickness is 99% chord then the original thickness is used to
% dispay for inpolygon use later on since we cannot plot a NACAXX100
% NACA_chosen is the original designation
if str2double(NACA_chosen(3))*10+str2double(NACA_chosen(4)) == 99

    [xi,zi] = panelgen(NACA_chosen,n,alpha);

else

    % Code to increase thickness of NACA by a tiny amount in order to make sure
    % flowfield never intersect with airfoil as long as max thickness isnt 99%
    % chord eg if NACA6409 for flowfield then NACA6410 will be used for
    % inpolygon
    thick_perc_inc = 1;

    % This code alows for the converting then reconverting of NACA as a string
    % to a double then back accounting for when its less than 10 and still
    % giving a 2 char string eg. '09'
    NACA = [NACA_chosen(1:2),num2str(((str2double(NACA_chosen(3:4)))+thick_perc_inc),'%02i')];
    [xi,zi] = panelgen(NACA,n,alpha);

end

% inpolygon() allows us to check our mesh against a polygon defined by our
% panelpoints with a boolean array output
% for the in array: 1 = in, 0 = out
% for the on array: 1 = on, 0 = not on
[in,on] = inpolygon(mesh_x,mesh_z,xi(1:n+1),zi(1:n+1));
```

Working out vertical and horizontal velocities at every point on grid from the chosen panel number

VARIABLES:

- **mesh_u -** Matrix containing total horizontal velocities for each point on grid
- **mesh_u -** Matrix containing total veritical velocities for each point on grid
- **utemp -** Matrix containing horizontal velocity produced by one panel at a time for each point on grid
- **vtemp -** Matrix containing veritical velocity produced by one panel at a time for each point on grid

With the inner loop from 1:N+1 and the ineer running from 1:stream_res^2 we can calculate how each individual panel affect every single point on our grid and then sum these temporary effects n+1 times to get the total sigma uij and vij form equation 8a and 8b

```
% Pre-assigning necessary arrays for future summation within for loops
[mesh_u,mesh_v,utemp,vtemp] = deal(zeros(size(mesh_x)));

% Calculating for each panel how it affects a specific point on grid
for i = 1:panel_number+1

    % Working out  how a specific panel affects all the points on grid
    for j = 1:stream_res^2

        [utemp(j),vtemp(j)] = cdoublet(([mesh_x(j),mesh_z(j)]),[x_panel_pts(i),z_panel_pts(i)],[x_panel_pts(i+1),z_panel_pts(i+1)]);

        % Summing  utemp and vtemp after multiplying by associated panel strength gives the total and final horizontal and vertical velocity for each point on
        mesh_u(j) = mesh_u(j)+n_panel_strengths(i)*utemp(j);
        mesh_v(j) = mesh_v(j)+n_panel_strengths(i)*vtemp(j);

    end

end

% Correcting velocity values of each point based on position relative to airfoil
% Using nan for anything in the airfoil shape to not display
mesh_u(in) = nan;
mesh_v(in) = nan;

% Adding velocities to every point that were imparted by the freestream
mesh_u(~in) = mesh_u(~in)+uinf*cosd(alpha);
mesh_v(~in) = mesh_v(~in)+uinf*sind(alpha);

% Using 0 for anything on the airfoil edge's shape as no slip condition
mesh_u(on) = 0;
mesh_v(on) = 0;
```

Adding Flowfield calculated from chosen panel number

VARIABLES:

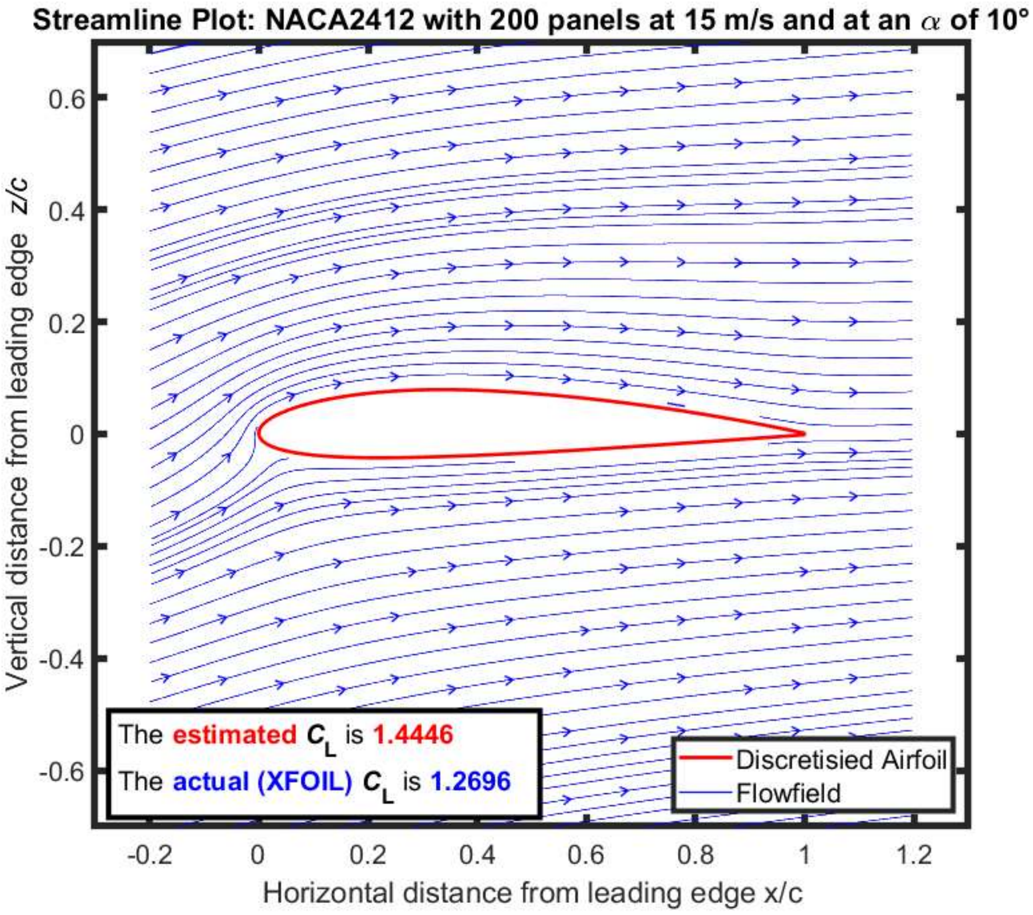
- **flowfield -** Plot for the streamlines of airflow
- **stream_legend -** Legend for plot
- **filetype -** File format for plot saving
- **quiver_plot -** Plot for quiver
- **quiver_ax -** Axes of quiver_plot
- **quiver_res -** Number of points along one side of Qiover grid
- **INT -** Array to select every quiver_res pt on grid
- **quiverfield -** Handle for quiver plot
- **quiv_textbox -** Annotation on quiver plot for Cl
- **quiv_child -** Children of quiver plot
- **quiv_xlabel** - Handle for xlabel of quiver plot
- **quiv_ylabel** - Handle for ylabel of quiver plot
- **quiv_title -** Handle for title of quiver plot

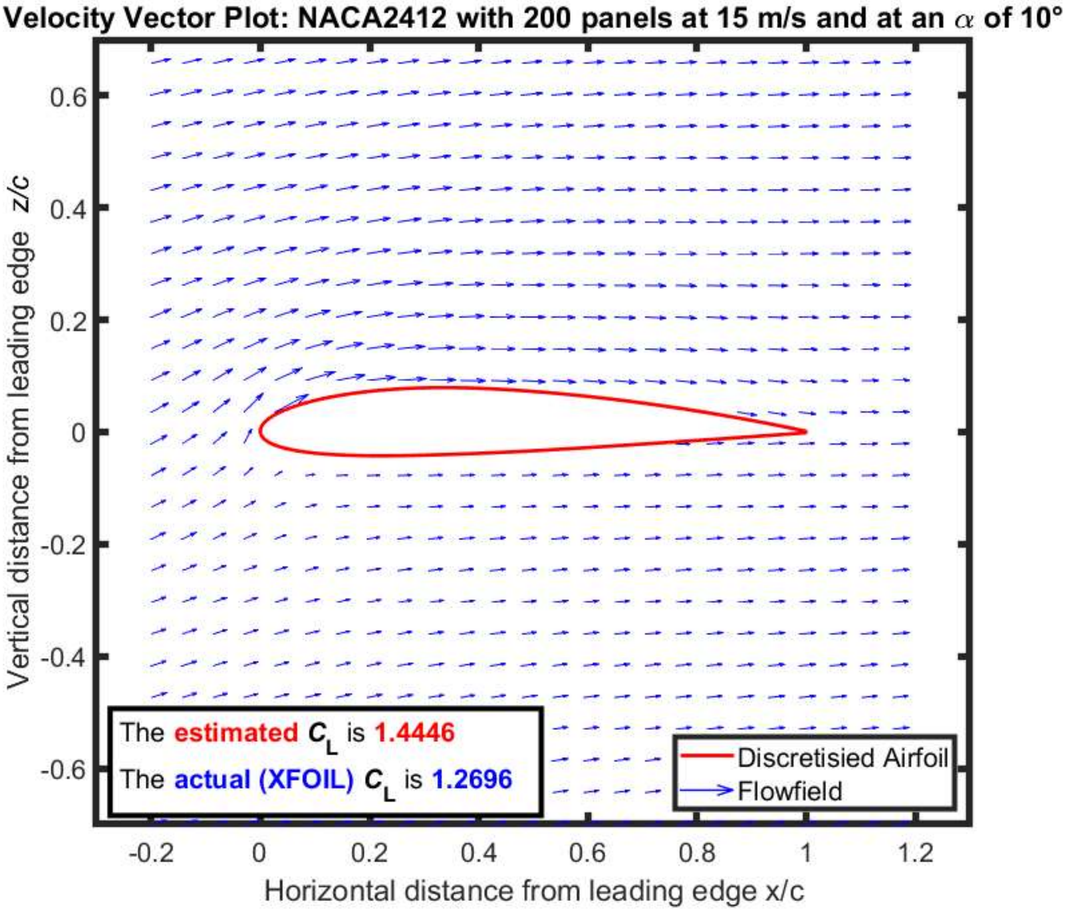
`_*disp_transform() is a function attached in the same folder as this script*_`

```
% Setting figure for quiver plot
% ('Color',[0.9,0.9,0.9]) is a selected colour of grey
quiver_plot = figure('Color',[0.9,0.9,0.9]);

% Using copyobj() to make copying panels from stream plot to quiver plot easily
quiver_ax = copyobj(stream_plot.CurrentAxes,quiver_plot) ;

% Setting number of points for length of grid of quiver to extract
quiver_res = 25;

% Setting a selection array to produce a similar plot to streamlines with less points
INT = round(1:stream_res/quiver_res:stream_res);

% Using quiver() to plot velocity vectors and indices (INT,INT) in order select an even distribution of quiver_res^2 points
quiverfield = quiver(mesh_x(INT,INT),mesh_z(INT,INT),mesh_u(INT,INT),mesh_v(INT,INT),'LineWidth',1,'Color','b','DisplayName','Flowfield');

% Streamslice() allows us to visualize flowfield with streamlines and arrows and 4 is the density of streamlines
figure(stream_plot)
flowfield = streamslice(mesh_x,mesh_z,mesh_u,mesh_v,4);

% Plot Styles with preset line widths from handout to produce clear plot
set(flowfield,'LineWidth',1,'DisplayName','Flowfield')

% If panel number is less than the critical number we must set the legend in different way
if panel_number < crit_panel

    % Stream plot legend now contains black (sharp) airfoil and smooth
    % (red) airfoil in legend
    stream_legend  =  legend([panels,smooth_airfoil,flowfield(1)],'Location','southeast','LineWidth',3,'FontSize',15,'TextColor','black','NumColumns',1);
    figure(quiver_plot)

    % Reordering legend by setting order of the lines using children of the
    % current axes
    quiv_child = quiver_plot.CurrentAxes.Children;
    quiv_legend = legend([quiv_child(3),quiv_child(5),quiv_child(1)],'Location','southeast','LineWidth',3,'FontSize',15,'TextColor','black','NumColumns',1);

else

    % Stream plot legend now only has smooth red airfoil
    stream_legend  =  legend([panels,flowfield(1)],'Location','southeast','LineWidth',3,'FontSize',15,'TextColor','black','NumColumns',1);
    figure(quiver_plot)

    % Reordering legend by setting order of the lines using children of the
    % current axes
    quiv_child = quiver_plot.CurrentAxes.Children;
    quiv_legend = legend([quiv_child(3),quiv_child(1)],'Location','southeast','LineWidth',3,'FontSize',15,'TextColor','black','NumColumns',1);

end

% Using dot notation and get() to set() the font size
quiv_xlabel=get(gca,'xlabel');
quiv_ylabel=get(gca,'ylabel');
quiv_title=get(gca,'title');
[quiv_xlabel.FontSize,quiv_ylabel.FontSize,quiv_title.FontSize]=deal(15);

% Re-assigning strings for title of each plot
quiv_title.String=[('Velocity Vector Plot: '),quiv_title.String];
stream_title=get(stream_plot.CurrentAxes,'title');
stream_title.String=[('Streamline Plot: '),stream_title.String];

% Initialising and then copying over CL annotation from stream_plot to
% quiver_plot
quiv_textbox=annotation('textbox','Visible','off');
copyobj(Cl_textbox,quiv_textbox.Parent)

% Setting the thickness of the axes and boundarys of the plot
set(quiver_plot.CurrentAxes,'linewidth',4)
set(stream_plot.CurrentAxes,'linewidth',4)

% Moving plots according to whether the 2nd cl/alpha plot will be produced e.g. if NACA is 2412
if NACA_chosen == num2str(2412)

    % These are arbitrary values that were used on original pc to create
    % the current visual
    stream_plot.Position=[disp_transform(853,285),disp_transform(843,720)];
```

```matlab
    quiver_plot.Position=[disp_transform(5,285),disp_transform(843,720)];

else

    % If not special case then move flowfield plot to centre
    % These are arbitrary values that were used on original pc to create
    % the current visual
    stream_plot.Position=[disp_transform(1269.666,233),disp_transform(1206,957)];
    quiver_plot.Position=[disp_transform(61,233),disp_transform(1206,957)];

end

% Final adjustment for fontsize on figures using dot notation and the
% screensize ratios
stream_plot.CurrentAxes.FontSize = stream_plot.CurrentAxes.FontSize*xratio;
stream_legend.FontSize = stream_legend.FontSize*xratio;
quiver_plot.CurrentAxes.FontSize = quiver_plot.CurrentAxes.FontSize*xratio;
quiv_legend.FontSize = quiv_legend.FontSize*xratio;

% Saving plot in same folder as script
% Time-stamping each plot as it saves
% clock() gets the date and time currently
% fix() allows for time to be displayed in hours mins and seconds
current_time = (fix(clock));
current_time = {current_time(1),current_time(2),current_time(3),current_time(4),current_time(5),current_time(6)};
clear str

if (current_time{4}) < 10

    % Because of way fix(clock) works adding a 0 when hours is only 1 digit
    % will allow for 24hr formatting
    current_time{4} = ['0',num2str(current_time{4})];

end

% Setting current_time as a string for easier use in saveas()
current_time = string(current_time);

% Allows for easy choice of file format for saving plots
filetype = 'png';


% Using num2str() and array indexing to extract hrs mins and seconds from
% current_time to timestamp file name
str = ['NACA',num2str(NACA_chosen),'_',num2str(panel_number),'n_',num2str(alpha),'deg_',num2str(uinf),'ms'];
str = [str,'__',current_time{3},'_',current_time{2},'_',current_time{1},'__at_'];
str = [str,current_time{4},'hr',current_time{5},'min',current_time{6},'sec.',filetype];
str_stream = ['Streamline_Plot_for_',str];
str_vect = ['Velocity_Vector_Plot_for_',str];

% Concatenating strings into final string that will beused as file name
saveas(stream_plot,str_stream)
saveas(quiver_plot,str_vect)

% Clearing workspace of unneeded variables
clear xi zi N_Panel_MUs NACA Cl n alpha uinf POSITION
```

```
The xf-naca2412-il-1000000.txt WAS READ.

The estimated coefficient of lift is 1.4446

The actual (XFOIL) coefficient of lift is 1.2696
```





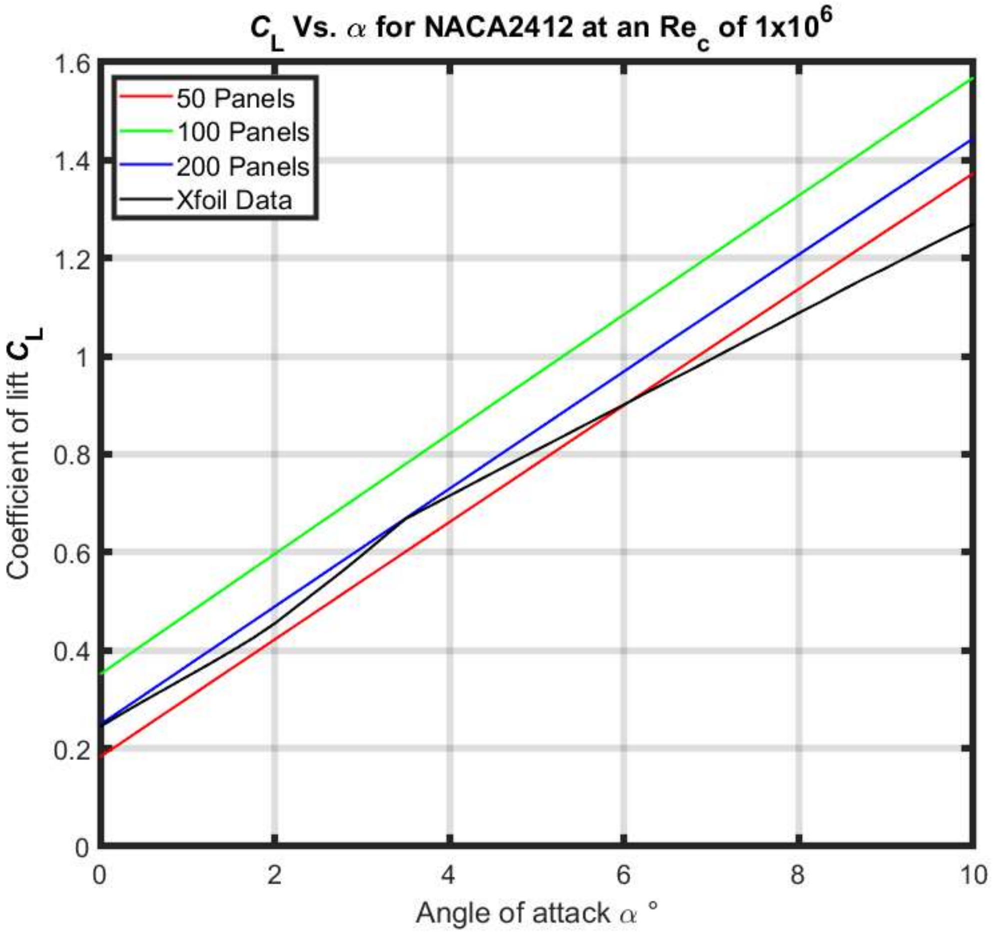
CL VS. ALPHA FOR NACA2412

VARIABLES:

- **Previously used variables** below have been explained previously

- **lowerbound -** Lowest value of angle of attack used for graph
- **upperbound -** Highest value of angle of attack used for graph
- **numberofpoints -** Number of points in data sets for calculated airfoils
- **Data -** Xfoil data converted into an array from the .txt file provided
- **alpha_xfoil** - Angles of attack for NACA2412 from Xfoil
- **Cl_xfoil** - Coefficients of lift for NACA2412 from Xfoil

```
% Only activating for special case
if NACA_chosen == num2str(2412)

    NACA = 2412;

    % Panel number and colour choice assigned in array to allow for counter in loop to define plot easily
    N = [50,100,200];
    colour = '-r-g-b-k';

    % Display size explained previously
    CL_ALPHAPLOT = figure('Color',[0.9,0.9,0.9],'Position',[disp_transform(1701,285),disp_transform(843,720)]);

    % Using a relatively small number as relationship will display as linear even at higher panel numbers but easily adjustable
    numberofpoints = 10;

    % Lower and upper bounds for angle of attack
    lowerbound = 0;
    upperbound = 10;

    % Reynolds Number is 1e6 in the xfoil data therefore assumption of
    % dynamic viscosity as 1.8e-5 and air density as 1.225to produce uinf
    uinf = 1e6*1.8e-5/1.225;

    % Pre-assigning arrays
    [alpha,Cl] = deal(zeros(numberofpoints+1,3));

    % Iterating over number of panels required
    for counter = 1:length(N)

        n = N(counter);
        % Pre-assigning arrays
        [xi,zi] = deal(zeros(1,n+2));

        % Calculating Cl with respect to alpha for each panel setting
        for count = 1:numberofpoints+1

            % alpha array gets created by adding increments onto lower
            % bound based on current count
            alpha(count,counter) = lowerbound-((upperbound-lowerbound)/numberofpoints)+count*((upperbound-lowerbound)/numberofpoints);

            % New airfoils must be generated to calculate new Cl
            [xi,zi] = panelgen(NACA,n,alpha(count,counter));
            [~,Cl(count,counter)] = finalcalc(xi,zi,uinf);

            %clearing for easy use in next iteration
            clear xi zi

        end

        % Plotting graph with styles
        plot(alpha(:,counter),Cl(:,counter),['-',colour(counter.*2)],'LineWidth',1.5,'DisplayName',[num2str(n),' Panels'])
        hold on

    end

    % XFOIL(find(XFOIL(:,1)==lowerbound):find(XFOIL(:,1)==upperbound),1) allows to find the exact Xfoil data that matches the angles we are working with
    alpha_xfoil(:,1) = XFOIL(find(XFOIL(:,1) == lowerbound):find(XFOIL(:,1) == upperbound),1);
    Cl_xfoil(:,1) = XFOIL(find(XFOIL(:,1) == lowerbound):find(XFOIL(:,1) == upperbound),2);

    % Plotting xfoil data and applying plot styles
    plot(alpha_xfoil(:,1),Cl_xfoil(:,1),['-',colour((counter+1)*2)],'LineWidth',1.5,'DisplayName','Xfoil Data')

    % Legend and setting thickness for borders
    CL_ALPHAPLOTLEGEND  =  legend('Location','northwest','FontSize',15,'LineWidth',3);
    set(gca,'Linewidth',4)
    xlim([lowerbound,upperbound])
    grid on

    % {\alpha} is latex format for alpha and char(176) is degrees
    title([bold_italic_black,' C','\rm',bold_black,'_{L}',' Vs. {\alpha} for ','NACA',num2str(NACA),' at an Re_{c} of 1x10^6'])
    xlabel(['Angle of attack {\alpha} ',char(176)])
    ylabel(['Coefficient of lift ',bold_italic_black,'C','\rm',bold_black,'_{L}'])

    % Final adjustment for fontsize on figures
    CL_ALPHAPLOT.CurrentAxes.FontSize = 15*xratio;
    CL_ALPHAPLOTLEGEND.FontSize = 15*xratio;

    % Saving plot
    % Time-stamping each plot as it saves
    clear str
    str = ['NACA',num2str(NACA_chosen),'__cl_vs_aoa__',current_time{3},'_',current_time{2},'_',current_time{1},'__at_'];
    str = [str,current_time{4},'hr',current_time{5},'min',current_time{6},'sec.',filetype];
    saveas(CL_ALPHAPLOT,str)

end
```



FINAL CLEAR

```
% Clearing variables to have a clear workspace whilst still displaying all requested information
clear

% Setting the warning back on for users
warning("on")
```

Published with MATLAB® R2021b