Preliminary Aerodynamic Study

Team 7:
Cooper LeComp, John A. Papas Dennerline,
Ian Greene, Christos Levy, Daniel Qi, Hans Suganda

18th September 2022

Contents

0.1 Rationale . . . 2
0.2 General Airfoil Comparisons . . . 2
0.2.1 "Fat" Airfoils . . . 2
0.2.2 "Thin" Airfoils . . . 2
0.3 Initial Airfoil Selection . . . 2
0.4 Airfoil Computational Analysis . . . 3
0.5 Algorithm Development . . . 4
0.5.1 Setup_Airfoil_Selection_Algorithm . . . 4
0.5.2 Fetch_Airfoils . . . 5
0.5.3 Produce_Airfoil_List . . . 6
0.5.4 Make_Airfoil_Directories . . . 6
0.5.5 Move_Airfoil_Directories . . . 6
0.5.6 Multi_Airfoil_Analysis . . . 7
0.5.7 Ave_CL_Max.py . . . 7
0.5.8 Single_Airfoil_Re_Analysis . . . 8
0.5.9 Gen_Input_File . . . 9
0.5.10 Comment_Out . . . 9
0.5.11 Read_Polar.py . . . 10
0.5.12 Plot_Airfoil_Geom.py . . . 10
0.5.13 Plot_CLMax_Re.py . . . 11
0.5.14 Plot_CL_Alpha.py . . . 11
0.6 Results of Computational Analysis . . . 12
0.7 Airfoil Experimental Analysis . . . 14
0.8 Wing Design . . . 14
0.8.1 Approximate Boundary Layer Thickness . . . 14
0.8.2 Aspect Ratio . . . 15
0.8.3 Tapered Wing Design . . . 15
0.8.4 Wing Design Algorithm . . . 16

0.1 Rationale

The airfoil is the heart of an aircraft. Selecting the correct airfoil allows us to predict the lift, drag, and pitching moments produced by the wing. The lift and drag is obviously important in flying the aircraft, and determining its cruising speed, orientation and power consumption. The pitching moments should be used in conjunction with the location of the center of mass to determine the sizing of the aircraft tail. Too small of an aircraft tail and there is not enough control authority, too large of a tail and the aircraft performance suffers.

0.2 General Airfoil Comparisons

0.2.1 "Fat" Airfoils

A "fat" airfoil is going to be defined as an airfoil with a large radius for its leading edge, is generally thick, and has a large amount of camber.

Advantages

So the large camber of the airfoil allows it to produce alot of lift at zero angle of attack. The large radius at its leading edge allows the airflow to remain attached, which increases the stall angle of the airfoil, allowing for a higher maximum lift coefficient. The correct thickness distribution is helpful in preventing flow separation. A thick airfoil has good stall characteristics. Stall will occur first in the trailing edge of the airfoil and progressively move to the leading edge. The loss of lift is gradual and pitching moments do not change by too much

Disadvantages

It has alot of drag. The induced drag is higher because of its lifting capabilities, but its viscous drag is also higher because it has more surface area due to its camber and thickness. The pitching moments about the aerodynamic center is also great due to the camber of the airfoil.

0.2.2 "Thin" Airfoils

A "thin" airfoil is going to be defined as an airfoil with a lower radius on its leading edge, is generally thin, and has a lower amount of camber.

Advantages

These airfoils can suppress turbulence, maintaining laminar flow and getting very good lift to drag ratios. These airfoils typically produce a low amount of lift and also produce very mild pitching moments.

Disadvantages

They cannot produce as much lift. If they are taken out of their laminar bucket, drag will increase and lift will fall quickly, this is because laminar flows are more prone to separation. They also exhibit terrible stall characteristics. Separation occurs complete and instantly, loss of lift is instant, and bad pitching moments will instantly appear during stall.

0.3 Initial Airfoil Selection

Based on the design requirements, the aircraft needs to have a stall speed that is as low as possible, which drives our wing area to be large. However, due to the geometric requirements of the craft, there is a limit to how large our wing area can be. So, if we make sure our wing produces as much lift as

possible, we can still have a reasonable wing area with a really low stall speed. So, we must prioritize high-lift capable wings. Therefore, we should choose "fatter" airfoils over their "thinner" counterparts.

Here is how we are going to choose our airfoils. We are going to have a list of airfoils just by visual inspection of which ones look "fat" for the high maximum lift coefficient and gentle stall characteristics. Then, we are going to run `xflr` on it and do wind-tunnel testing to rule all the others out and choose the one that best suits our needs. The selected airfoils are:

Airfoil Name	Max Thickness	Max Thick Loc	Max Camber	Max Camber Loc
Eppler 420	14.3	22.8	10.6	40.5
Eppler E423	12.5	23.7	9.5	41.4
S1223	12.1	19.8	8.1	49.0
S1223 RTL	13.5	19.9	8.3	55.2
Chuch Hollinger CH 10-48-13	12.8	30.6	10.2	49.3
Curtiss CR-1	12.2	24.0	4.7	42.0
Drela DAE11	12.8	32.8	6.6	44.4
Drela DAE21	11.8	32.0	6.6	43.7
Drela DAE31	11.1	29.3	6.7	47.0
Lissaman 7769	11.0	30.0	4.4	30.0
Naca 2412	12.0	30.0	2.0	40.0
N-24	17.0	30.0	5.4	40.0
Naca 6412	12.0	30.1	6.0	39.6
Naca 2414	14.0	29.5	2.0	39.6
Eppler E 395	12.3	29.5	5.2	52.3

0.4 Airfoil Computational Analysis

$$R = \frac{\rho U L}{\mu}$$

Taking density of air to be $\rho = 1.225\, kgm^{-3}$, dynamic viscosity $\mu = 1.81 \times 10^{-5}\, kgm^{-1}s^{-1}$. Our stall speed was prescribed by the requirements to be around $V_{stall} = 4.572\, ms^{-1}$. Our chord length is predicted to be around $0.12\, m$ at the smallest part of the wing so to be safe, let us try to simulate for chord length of around $0.05\, m$. The `Python` script below is used to compute the Reynold's number,

```
#!/bin/python
def R_Num(U, L):
    '''U here is the velocity, enter stall speed
    L here is the length, enter target chord length'''
    rho = 1.225 #Density in kg/m^3
    mu = 1.81*10**-5 #kgm^{-1}s^{-1}
    R = (rho*U*L)/mu #Compute Reynold's Number
    return R
```

The results of running the algorithm for the stall speed and minimum chord length gives the lower bound Reynold's number that the `xflr` simulations ought to be run at,

```
>>> R_Num(4.572, 0.05)
15471.546961325967
```

Running the algorithm for the same stall speed but at the predicted maximum chord length of around $0.4\, m$ gives us the upper bound Reynold's number that the `xflr` simulations should be run at,

```
>>> R_Num(4.572, 0.4)
123772.37569060773
```

For extra insurance, let us run our `xflr` analysis on the range of .

$$10000 \leq R_e \leq 150000 \tag{1}$$

0.5 Algorithm Development

Below describes the theoretical algorithmic flow:

1. Given an airfoil, we are going to run for a series of Reynold's number ranges prescribed based on equation 1.
2. For each of those Reynold's number, we are going to run simulations for a range of angles of attack. The range of the angles of attack is going to be high enough such that we can figure out the stall angle of attack and the corresponding maximum lift coefficient.
3. Now each airfoil for differing Reynold's number is going to have different stall angles and different corresponding maximum lift coefficient. Stall is a separation effect and it is a very bad regime to fly in. Therefore, we want to choose flight regimes without stall. This is just the limit of the airfoil.
4. This means that for a given airfoil, amongst all the different maximum lift coefficient due to Reynod's number variation, we will choose the minimum $c_{L,max}$.
5. We will also compute the average maximum lift coefficients for a given airfoil amongst its differing Reynold's number ranges justto compare that the minimum $c_{L,max}$ is not due to results that have failed to converge.

The simulation will require extensive file manipulations such as renaming, moving files around, and requires the interaction of many programs. Therefore, the language of choice for this simulation would be the `Bash`. We are going to present a series of `Bash` scripts that is used to automate testing and work. There will be a few data analysis section and for that we decided to use `Python` due to its ease and extensive data-related libraries.

This is how the algorithm's directory tree is structured. All of the related `Bash` and `Python` scripts will be placed in only a single directory. Within that directory where all of the source code resides, there will be just a single directory that contains all of the Airfoil-specific data and this directory is creatively named `Airfoil_Data`. Let us first go through how the files are prepared.

0.5.1 Setup_Airfoil_Selection_Algorithm

The `Bash` script below is used to fetch all of the necessary airfoil data from the web databases and store the airfoil coordinates in individual directories. These individual directories will then be placed inside the data directory named `Airfoil_Data`. Let us examine the various utilities called from within this shell script,

```
#!/bin/bash

Airfoil_Data_Directory="Airfoil_Data"
mkdir $Airfoil_Data_Directory

#Fetch all of the Airfoil Data
./Fetch_Airfoils
```

```
#Produce a List of all the Airfoils Available for sim
./Produce_Airfoil_List

#Produce All of the Airfoil Directories
./Make_Airfoil_Directories Airfoil_List.txt

#Move All Airfoil Directories Out
./Move_Airfoil_Directories Airfoil_List.txt $Airfoil_Data_Directory
```

Note to run the set-up shell script, it is very easy, just call the following command. Everything else should be handled for you after calling that command.

```
./Setup_Airfoil_Selection_Algorithm
```

0.5.2 Fetch_Airfoils

Ok, so the first utility called by the script shown in section 0.5.1 is `Fetch_Airfoils`. I mentioned about fetching data automatically from web-servers right? Well this is exactly what the script does! `wget` is a common `Linux` utility that allows the download of a certain file from the web. So the way `wget` is used in this instance is like this:

```
wget WEB_ADDRESS -O NAME_FILE
```

For different airfoils, we just change the argument `WEB_ADDRESS` with where we believe the airfoil coordinate file is located in the internet. We would also want to give the downloaded file a good name we like. For that, we change the argument `NAME_FILE`. Anyways, this is not a tutorial on how to use `wget`, more details can be found in its official site documentation. For now, the script below fetches all of the airfoi cordinates in selig format from different web-addresses and renames them.

```
#!/bin/bash

#Indicate Start of Download
echo "Starting␣to␣Fetch␣Airfoil␣Data..."

# For Airfoil Eppler 420
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=e420-il -O Eppler_420.dat

# For Airfoil Eppler E423
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=e423-il -O Eppler_E_423.dat

# For Airfoil S1223
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=s1223-il -O S1223.dat

# For Airfoil S1223 RTL
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=s1223rtl-il -O S1223_RTL.dat

# For Airfoil Chuch Hollinger
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=ch10sm-il -O Church_Hollinger.dat

# For Airfoil Curtiss CR-1
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=cr1-il -O Curtiss_CR1.dat

# For Airfoil Drela DAE11
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=dae11-il -O Drela_DAE11.dat

# For Airfoil Drela DAE21
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=dae21-il -O Drela_DAE21.dat

# For Airfoil Drela DAE31
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=dae31-il -O Drela_DAE31.dat
```

```
# For Airfoil Lissaman 7769
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=l7769-il -O Lissaman_7769.dat

#For Airfoil Naca_2412
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=naca2412-il -O Naca_2412.dat

#For Airfoil N-24
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=n24-il -O N-24.dat

#For Airfoil Naca_6412
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=naca6412-il -O Naca_6412.dat

#For Airfoil Naca_2414
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=n2414-il -O Naca_2414.dat

#For Airfoil Eppler 395
wget http://airfoiltools.com/airfoil/seligdatfile?airfoil=e395-il -O Eppler_E_395.dat

#Customary End
echo "Finished␣Fetching..."
```

0.5.3 Produce_Airfoil_List

So after running `Fetch_Airfoils` we are going to have a bunch of files named `Eppler_Airfoil_whatever.dat`, `Another_Random_Airfoil.dat` and so on for all of the airfoils we listed in the script above. So now it would be convenient if we have a file containing just the names of the airfoils. This file would then be used in naming the various directories that the airfoil coordinate files would be placed in.

```
#!/bin/bash

Airfoil_List_File_Name=Airfoil_List.txt
ls *.dat > $Airfoil_List_File_Name
sed -i s/.dat//g $Airfoil_List_File_Name
```

0.5.4 Make_Airfoil_Directories

This `Bash` script takes an input file as an argument and then goes through the contents of that file, making new directories corresponding to each airfoil inside the given input file and then putting the airfoil coordinate data into the newly-created directory.

```
#!/bin/bash

while read Airfoil_Name; do
    mkdir $Airfoil_Name
    mv $Airfoil_Name".dat" $Airfoil_Name"/"
done < $1
```

0.5.5 Move_Airfoil_Directories

This `Bash` script has a simple task, move all the directories found inside a file list to a new location that is given by the second input argument. This is used to move all of the newly created directories corresponding to each airfoil into a directory named `Airfoil_Data`.

```
#!/bin/bash

while read Airfoil_Name; do
    mv $Airfoil_Name/ $2/
done < $1
```

0.5.6 Multi_Airfoil_Analysis

This is the next `Bash` script that needs to be explained. This `Bash` script is the allows us to perform analysis on multiple airfoils in one single run. After we have done all of the initial setup described in section 0.5.1, we can run the airfoil analysis by invoking the command,

```
./Multi_Airfoil_Analysis Airfoil_List.txt
```

wherein `Airfoil_List.txt` is a text file containing all of the airfoil names mentioned previously. `Multi_Airfoil_Analysis` depends on a lot of other scripts. So the script will run all the Reynold's Number simulations and stall conditions for each listed airfoil, collect all of the data from all of the listed airfoils and output the summary in a file named `Raw_Multi_Airfoil_Analysis.tex`.

```
#!/bin/bash

#Arg1: File containing all of the airfoil_data

#This script collects all of the output files for eachh airfoil and figures out:
#Minimum cl_max, Average cl_max, How many Data did  reach stall

Initial_Re_N=10000
Final_Re_N=150000
Re_N_Inc=5000

Multi_Airfoil_Output_File="Raw_Multi_Airfoil_Analysis.tex"

#Print out the header
echo "#|Airfoil␣Data␣Name|Minimum␣CL_Max|␣Average␣CL_Max|␣Number␣of␣Valid␣Data|Number␣of␣Total␣Data|
    " >> $Multi_Airfoil_Output_File

#Loop over the Input File
while read Airfoil_Name; do
    #Definition of Additional Address
    Add_Address="Airfoil_Data/"$Airfoil_Name
    #Run the Simulation for a single airfoil and a Reynold's Number Range
    ./Single_Airfoil_Re_Analysis $Airfoil_Name $Initial_Re_N $Final_Re_N $Re_N_Inc
    #Print out the Airfoil Name for a single entry
    echo -n $Airfoil_Name"␣">>$Multi_Airfoil_Output_File
    #Print out the Relevant Data for each entry
    ./Ave_CL_Max.py $Add_Address/$Airfoil_Name"_Multi_Re.tex">>$Multi_Airfoil_Output_File
done < $1
```

0.5.7 Ave_CL_Max.py

We are counting on the `Single_Airfoil_Re_Analysis` to basically generate a file corresponding to each airfoil containing the various Reynold's numbers the particular airfoil was tested at as well as the maximum lift coefficient. This `Python` script operates on that file, gets rid of all of the invalidated data where stall conditions were not met or poor convergence was obtained and then using the valid data points, figure out the average maximum lift coefficient and the smallest lift coefficient for that airfoil across all of its Reynold's number ranges.

```
#!/bin/python

#Library imports
import numpy as np
import sys

Input_File = str(sys.argv[1])

#Read in the data file
data = np.genfromtxt(Input_File)
```

```
#How much data points were given
Total_N = np.shape(data)[0]

#Indices of which entries are valid
valid_entries = data[:,2]>0

#How many data points are valid
Valid_N = np.shape(data[valid_entries,0])[0]

#Computation of Average Maximum Lift Coeff due to valid data points
CL_Max_Ave = np.mean(data[valid_entries, 1])

#Computation of Minimum cl_Max
CL_Max_Min = np.min(data[valid_entries, 1])

#Print out all of the various data to stdout
print(CL_Max_Min, CL_Max_Ave, Valid_N, Total_N)
```

0.5.8 Single_Airfoil_Re_Analysis

Now that we have explained the way multiple airfoils are considered, we need to examine how individual airfoils are considered at a time. This `Bash` script is responsible for calling a few functionalities which would be explained in more detail.

```
#!/bin/bash

#Arg1: Airfoil Data Name
#Arg2: Initial Reynold's Number
#Arg3: Final Reynold's Number
#Arg4: Increment of Reynold's Number

#Appending To Path
Additional_Address=Airfoil_Data/$1

#Name of the output File
Output_File=$Additional_Address/$1_Multi_Re.tex

#Angle of Attack Variables
AOA_Init=0
AOA_Final=50
AOA_Inc=0.1

#Print the Name of the Airfoil
echo "# Airfoil Name: "$1 > $Output_File

#Print the Data Header of the airfoil Re, Max_CL, stall reached or not
echo "# |Reynold's Number | Maximum Lift Coeff | Stall Reached? (y=1,n=0)|" >> $Output_File

#Iterate Over Each Reynold's Number
for ((Re = $2; Re <= $3; Re=Re+$4)) ;do
    #Name of XFoil Output File
    XFoil_Output_File=$Additional_Address/$1_Output_RE_$Re.tex
    #Generate the Input script for the Reynold's number and AOA and dat file
    ./Gen_Input_File $1 $Re $AOA_Init $AOA_Final $AOA_Inc > $Additional_Address/Temp_Instruct.txt
    #Go to where the airfoil data is located
    cd $Additional_Address/
    #Run XFoil with the generated Input script
    xfoil < Temp_Instruct.txt
    #Delete the Instruction Script
    rm Temp_Instruct.txt
    #Jump back to where you were
    cd ../../
    #Comment Out the resulting data produced by XFoil so Python can read
```

```
    ./Comment_Out $XFoil_Output_File
    #Write Reynold's Number to Output File
    echo -n $Re" " >> $Output_File
    #Write max cl and stall status to Output File
    ./Read_Polar.py $XFoil_Output_File >> $Output_File
done
```

0.5.9 Gen_Input_File

We have decided to interface with `Xfoil` directly using the terminal. To interface with `Xfoil` we need to give it command instructions. Now the script shown below generates the `XFoil` specific instructions needed to run simulations at a desired Reynold's number and a range of angle of attack.

```
#!/bin/bash

#Arg1: Airfoil Data File
#Arg2: Reynold's Number
#Arg3: First AOA
#Arg4: Last AOA
#Arg5: Increment of AOA's

#Disable Plot Pop-ups
echo -e "plop\nG, F\n"

#To load the airfoil data we can use the following command
echo "load $1.dat"

#Enter operation mode
echo "oper"

#Change Iteration Solvers
echo "ITER 2000"

#Set the Reynold's Number
echo "Visc $2"

#Set the Mach Number
echo "Mach 0"

#Set the type of simulation we seek
echo "Type 1"

#Save the polars
echo -e "Pacc\n\n"

#Run the angles of attack
echo "aseq $3 $4 $5"

#Save the polars to a file
echo -e "PWRT\n$1_Output_RE_$2.tex "

#Exit to the main menu
echo -e "\n"

#Quit Xfoil
echo "Quit"
```

0.5.10 Comment_Out

After `Xfoil` has completed running, we need to comment the first few lines with `#` so that `Python` can easily read in and analyze the data using `numpy`. This script does that. It adds the character `#` into

the first 12 lines of a specified file.

```
#!/bin/bash

sed -e "1,12s/^/#/" $1 > Temp.txt
mv Temp.txt $1
```

0.5.11 Read_Polar.py

This is another `Python` script that operates on a single file that `Xfoil` generates. `XFoil` is going to generate lift coefficients and angles of attack at a particular Reynold's number. This `Python` script checks if the simulation run in `xfoil` has indicated the presence of a stall and what is the maximum lift coefficient at that stall angle.

```
#!/bin/python

#Library Imports
import numpy as np
import sys

#Arg1: The name of the data file
Data_File_Name = str(sys.argv[1])

#Read in the given data file
data = np.genfromtxt(Data_File_Name)

#What to do if data is empty
if (np.shape(data)[0] and (np.ndim(data) != 1)):

    #Figure out the maximum lift coefficient
    max_CL = np.max(data[:,1])

    #Print out the maximum lift coefficient
    print(max_CL ,end='␣')

    #Figure out if stall angles have been achieved
    if (max_CL > data[-1,1]):
        print('1')
    else:
        print('0')
else:
    print(0,'␣',0)
```

0.5.12 Plot_Airfoil_Geom.py

Scripts which are named starting with `Plot` are basically just diagnostic tools. These are scripts used only to examine the data and make give a clearer picture of the behaviour of these airfoil simulations. This particular script plots the airfoil geometry. It might be useful to examine if the panels are sufficiently high resolution or if further refinemenet is necessary.

```
#!/bin/python

#Library imports
import matplotlib.pyplot as plt
import numpy as np
import sys

file = str(sys.argv[1])
data = np.genfromtxt(file)

#Plotting Time History of AOA
```

```
figure_Airfoil, axis_Airfoil = plt.subplots()
axis_Airfoil.plot(data[:,0], data[:,1], 'kx-',label='Airfoil')
axis_Airfoil.set_xlabel('x-coordinates')
axis_Airfoil.set_ylabel('y-coordinates')
axis_Airfoil.set_title('Airfoil Geometry')
axis_Airfoil.legend()
axis_Airfoil.grid()
axis_Airfoil.axis('equal')

plt.show()
```

An example of using the script is shown below,

```
./Plot_Airfoil_Geom.py Airfoil_Data/S1223/S1223.dat
```

The command above would plot the `S1223` airfoil coordinate files.

0.5.13 Plot_CLMax_Re.py

Scripts which are named starting with `Plot` are basically just diagnostic tools. We know that for different Reynold's Numbers, the maximum lift coefficient is going to vary. This `Python` script plots the relationship between the maximum lift coefficient as a function of Reynold's number for just one particular airfoil. It is very useful to tell anomalous results occuring in which Reynold's Number cases.

```
#!/bin/python

#Library imports
import matplotlib.pyplot as plt
import numpy as np
import sys

file = str(sys.argv[1])
data = np.genfromtxt(file)
data = data[data[:,2]>0]

#Plotting Time History of AOA
figure_CL_Max, axis_CL_Max = plt.subplots()
axis_CL_Max.plot(data[:,0], data[:,1], 'rx',label='Airfoil')
axis_CL_Max.set_xlabel('Reynold\'s Number')
axis_CL_Max.set_ylabel('Maximum Lift Coefficient')
axis_CL_Max.set_title('Lift Vs Reynold\'s Number')
axis_CL_Max.legend()
axis_CL_Max.grid()

plt.show()
```

An example of using the script above to plot the maximum lift coefficient with Reynold's Number for a particular airfoil,

```
./Plot_CLMax_Re.py Airfoil_Data/Eppler_E_423/Eppler_E_423_Multi_Re.tex
```

The command above would plot the maximum lift coefficient vs Reynold's Number for the `Eppler_E_423` airfoil.

0.5.14 Plot_CL_Alpha.py

Scripts which are named starting with `Plot` are basically just diagnostic tools. This particular script plots lift coefficient as a function of angle of attack α. Now this script only works for a single airfoil for a single Reynold's number case.

```
#!/bin/python

#Library imports
import matplotlib.pyplot as plt
import numpy as np
import sys

file = str(sys.argv[1])
data = np.genfromtxt(file)

#Plotting Time History of AOA
figure_CL, axis_CL = plt.subplots()
axis_CL.plot(data[:,0], data[:,1], 'rx',label='Airfoil')
axis_CL.set_xlabel('Angle␣of␣Attack␣(degrees)')
axis_CL.set_ylabel('Lift␣Coefficient')
axis_CL.set_title('Lift␣Slope␣of␣Airfoil')
axis_CL.legend()
axis_CL.grid()

plt.show()
```

An example of using the script above to plot the lift coefficient as a function of angle of attack for a particular airfoil at a particular Reynold's number is shown below,

```
./Plot_CL_Alpha.py Airfoil_Data/Church_Hollinger/Church_Hollinger_Output_RE_30000.tex
```

The command above would plot the lift coefficient vs angle of attack for the `Church Hollinger` airfoil at a Reynold's Number of 30000.

0.6 Results of Computational Analysis

From the 10 initial airfoils chosen, here are the tabulated results,

```
#|Airfoil Data Name|Minimum CL_Max| Average CL_Max| Number of Valid Data|Number of Total Data|
Church_Hollinger    0.5932  1.5153933333333331  15  29
Curtiss_CR1         0.7187  1.3507095238095237  21  29
Drela_DAE11         1.4847  1.49895             6   29
Drela_DAE21         1.0051  1.4635125000000002  8   29
Drela_DAE31         1.4984  1.5358555555555558  9   29
Eppler_420          0.5736  1.4463708333333336  24  29
Eppler_E_423        1.6165  1.624511111111111   18  29
Lissaman_7769       0.8487  1.06074             10  20
S1223               0.024   1.748767857142857   28  29
S1223_RTL           0.1112  1.5654076923076923  26  29
N-24                0.6751  1.4441823529411766  17  29
Naca_2412           1.1216  1.2042000000000002  7   29
Naca_2414           0.6722  1.1293666666666666  6   29
Naca_6412           1.5899  1.5960666666666665  6   19
Eppler_E_395        0.8721  1.47034             15  29
```

As one can see, there are several convergence issues with `Xfoil` for some of the airfoils. Initially, the `Selig 1223` airfoil and the `Eppler E 423` airfoils look like good candidates, but structurally the bent airfoils may not be ideal for manufacturing. They also tend to stall at a lower angle of attack, which makes the aircraft more sensitive to the flight conditions. For these reasons, we have decided to move on with the `NACA 6412` airfoil.

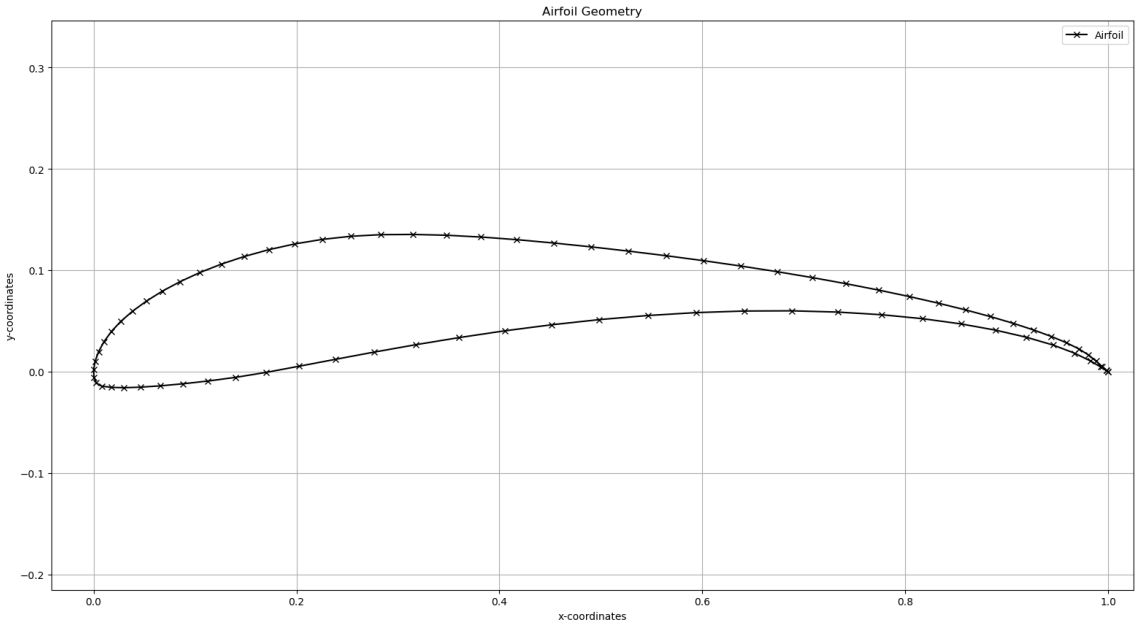

Figure 1: Representation of the Selig 1223 Airfoil

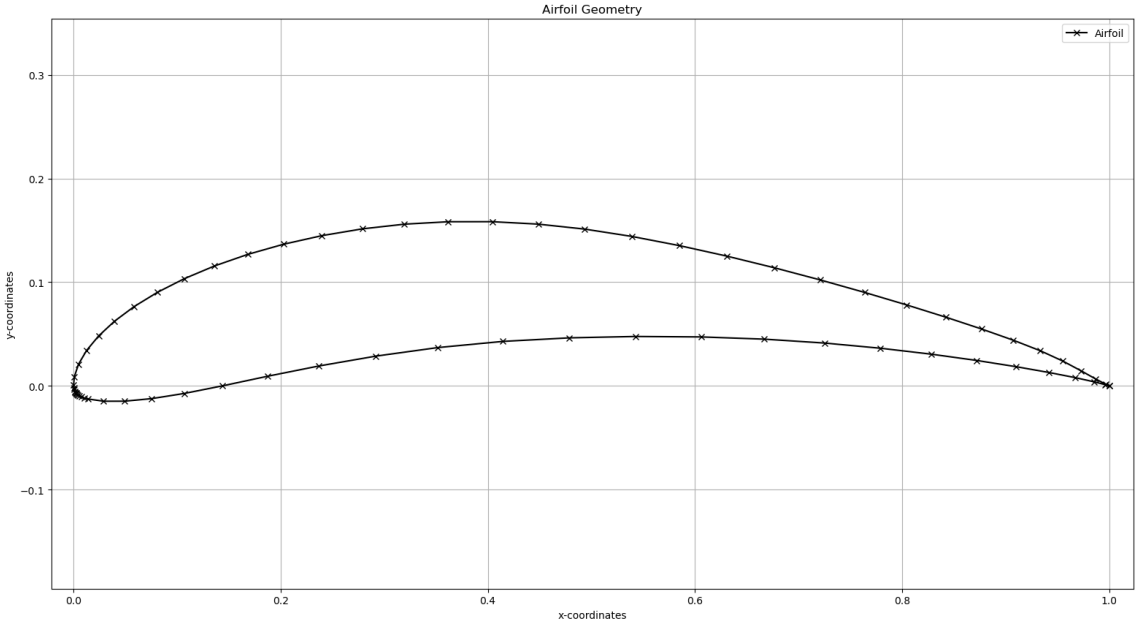

Figure 2: Representation of the Eppler E 423 Airfoil

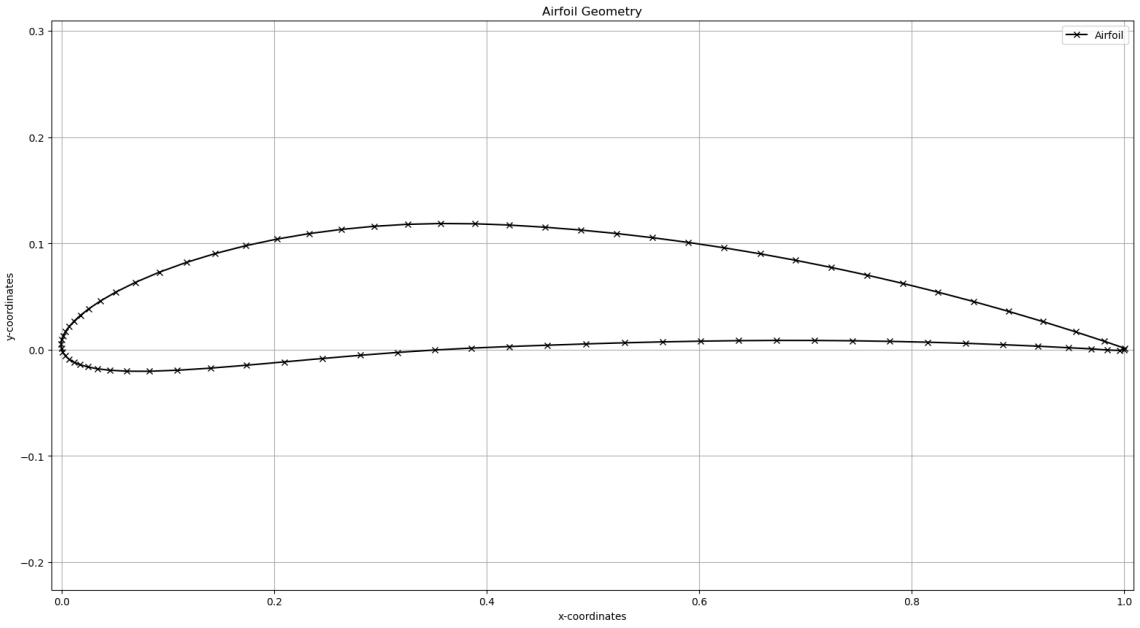

Figure 3: Representation of Naca 6412 Airfoil

0.7 Airfoil Experimental Analysis

We will have to conduct further wind-tunnel testing to verify the performance of our airfoils.

0.8 Wing Design

0.8.1 Approximate Boundary Layer Thickness

Having a way to estimate the thickness of the boundary layer is useful in the case wherein we need to make vortex generators around the control surfaces. The vortex generators have to be around the size of the boundary layer as to ensure its effectiveness and that is preturbs the free-stream as minimally as possible.

Here, the boundary layer is assumed to be laminar. Laminar boundary layer thickness δ can be estimated by the following relation,

$$\frac{\delta}{x} = \frac{5}{\sqrt{R(x)}}$$

Substituting for the Reynold's number based on length x,

$$R(x) = \frac{\rho U_\infty x}{\mu}$$

Attempting to substitute Reynold's number as a function of x,

$$\delta = \frac{5x}{\sqrt{R(x)}} = 5x\frac{1}{\sqrt{R(x)}} \tag{2}$$

We would need an expression for the square root reciprocal of Reynold's number. Taking the recpriprocral of the Reynold's number first,

$$\frac{1}{R(x)} = \frac{\mu}{\rho U_\infty x}$$

Taking the square root of this recpriocral,

$$\frac{1}{\sqrt{R(x)}} = \sqrt{\frac{\mu}{\rho U_\infty x}}$$

Substituting this into equation 2,

$$\delta = 5x\frac{1}{\sqrt{R(x)}}$$

$$\delta = 5x \times \sqrt{\frac{\mu}{\rho U_\infty x}} == 5 \times \sqrt{\frac{\mu x^2}{\rho U_\infty x}} = 5\sqrt{\frac{\mu x}{\rho U_\infty}}$$

0.8.2 Aspect Ratio

The expression for aspect ratio is shown below,

$$A_r = \frac{b^2}{A_w} \tag{3}$$

wherein b represents the span of the aircraft wing and A_w represents the area of the aircraft wing. We can re-manipulate this equation to obtain the span of a given wing,

0.8.3 Tapered Wing Design

Due to constraints in our manufacturing process, a tapered wing seems to suit our needs. We also choose a taper ratio of $\lambda = 0.45$ which would be closest to the elliptical lift distribution. The taper ratio for a wing is shown below,

$$\lambda = \frac{L_{c,t}}{L_{c,r}} \tag{4}$$

wherein $L_{c,r}$ represents the chord length of the wings at the roots and $L_{c,t}$ represents the chord length of the wings at the tips. A tapered wing is essentially a trapezoid. Therefore, its area could be described by the expression below,

$$A_w = \frac{1}{2}b(L_{c,r} + L_{c,t})$$

We would want to express the wing area in terms of the root chord length. Let us use equation 4 to express $L_{c,t}$ in terms of $L_{c,r}$.

$$\lambda = \frac{L_{c,t}}{L_{c,r}}$$

$$\lambda L_{c,r} = L_{c,t}$$

Substituting to the wing area equation,

$$A_w = \frac{1}{2}b(L_{c,r} + \lambda L_{c,r})$$

$$A_w = \frac{1}{2}bL_{c,r}(1 + \lambda)$$

$$A_w = \frac{b(1+\lambda)}{2}L_{c,r}$$

We would like to get an expression for the span of the wing though,

$$\frac{A_w}{L_{c,r}} = \frac{b(1+\lambda)}{2}$$

$$\left(\frac{A_w}{L_{c,r}}\right)\frac{2}{(1+\lambda)} = b \tag{5}$$

0.8.4 Wing Design Algorithm

We are going to assume the input to this to be the total main wing area and the root chord of the main wing. The total main wing area was obtained from knowing the wing loading of the aircraft and approximate sizing determined earlier. The root chord of the wing is a sane parameter because we need to make sure that the wing root can "fit" into the fuselage correctly.

1. We can use equation 5 to get the span of the wing from the root chord and the total main wing area.

2. We are then going to compute the aspect ratio of the wing based on equation 3. This an important parameter, and we might use it later.

The `Python` script below implements the equations discussed above to arrive at the wing parameters,

```
#!/bin/python

#Variable declarations
A_w = 0.903224 #Wing Area (m^2)
L_cr = 0.3556 #Wing Root Chord (m)
lambda_w = 1.0 #Taper Ratio

#Compute Tip Wing Chord
L_ct = L_cr*lambda_w

#Compute Span from Wing Area and wing Chord
b = (A_w/L_cr)*(2.0/(1.0 + lambda_w ))

#Compute Aspect Ratio of Wing
A_r = (b**2)/A_w

#Show the Wing Design
HED="#Wing␣Area␣(m^2)|Taper␣Ratio|Wing␣Root␣Chord␣(m)|Tip␣Root␣Chord␣(m)|Wing␣Span␣(m)|Aspect␣Ratio"
print(HED)
print(A_w, lambda_w, L_cr, L_ct, b, A_r)
```

The result of running the algorithm is shown below,

```
#Wing Area (m^2)|Taper Ratio|Wing Root Chord (m)|Tip Root Chord (m)|Wing Span (m)|Aspect Ratio
0.903224 1.0 0.3556 0.3556 2.54 7.142857142857142
```

Based on extensive discussions, we have decided not to use a tapered wing to maximize wing area and reduce stall speed. The main wing's airfoil cross-section is the `Naca 6412`, and it will also be $0.903224\,m^2$ in area, $0.3556\,m$ in root chord with no taper. The tail would use a symmetric airfoil `Naca 0012` with no taper and an aspect ratio of 5.