Airfoil data analysis & interpretation

AE102 Project

Airbenders - Group 3

Team members
Chaitanya Shankar Moon - 200010015
Meera Upasana - 190110046
Sahil Sanjay Vaidya - 200010065
Chauhan Tejaswini Ramdas - 200010017
Siddhant Gedam - 200010077
Kushagra Mishra - 200010042

Guides: Prof. Amuthan A. Ramabathiran & Prof. Prabhu Ramachandran

Contents

1 Introduction

1.1 Motivation for topic

We were very excited to get hands-on experience on data analysis on a topic of our choice. For choosing the topic, we found two interesting ideas- One was doing Bitcoin analysis and other was to do something related to Aerospace. Eventually, our love for aerospace made us choose 'Airfoil data analysis' as our topic. This gave us a chance to combine our learnings from AE152 and AE102 and apply it to real world data.

1.2 About Airfoils

An **airfoil** is the cross-sectional shape of a wing; blade of a propeller rotor or turbine; or sail as seen in cross-section.
A solid body moving through a fluid produces an aerodynamic force. The component of this force perpendicular to the relative freestream velocity is called **lift**. The component parallel to the relative freestream velocity is called **drag**. An airfoil is a streamlined shape that is capable of generating significantly more lift than drag. Subsonic flight airfoils have a characteristic shape with a rounded leading edge, followed by a sharp trailing edge, often with a symmetric curvature of upper and lower surfaces.

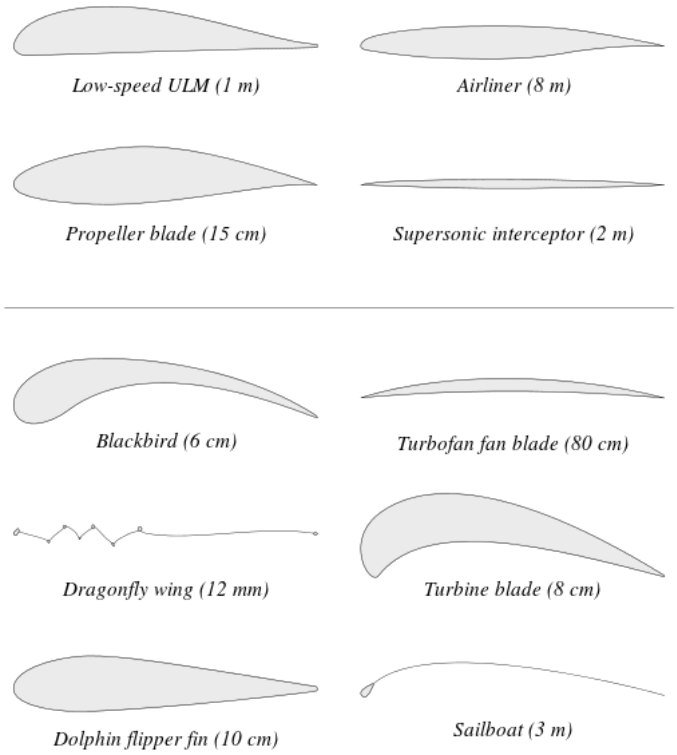


Figure 1: Different airfoils- Man-made and naturally occurring

The equation for lift (L) involving freestream velocity(V_∞), density(ρ_∞) and surface area of wing (S) is given by,

$$L = \frac{\rho_\infty V_\infty^2 S C_L}{2}$$

with C_L being lift coefficient.

In a similar way, equation for drag (D) is given by,

$$D = \frac{\rho_\infty V_\infty^2 S C_D}{2}$$

with C_D being drag coefficient.

The angle that the chord line of an airfoil makes with freestream velocity is called Angle of attack, denoted by α. It is empirically known that α and C_L have a linear relationship before separation of flow (stalling) starts, as shown in figure below.

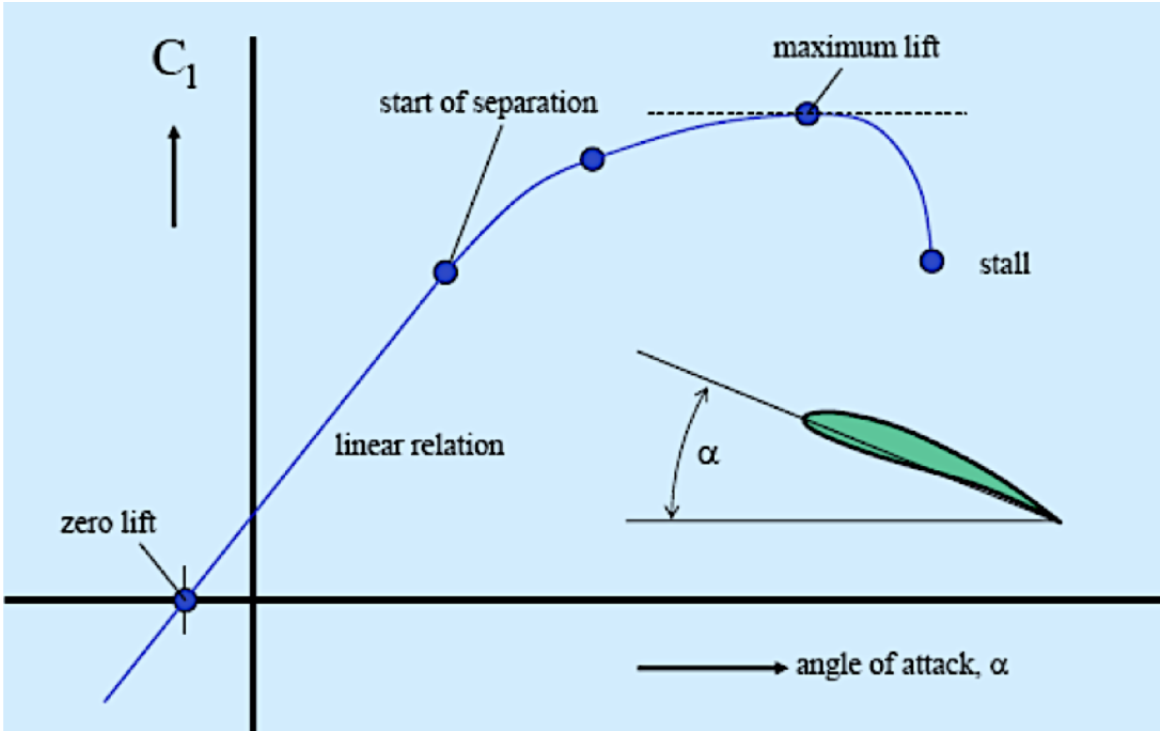


Figure 2: C_L vs α curve

Further, empirical relation between C_L and C_D for a finite wing is known to be of quadratic nature. The equation is as follows-

$$C_D = C_{D,min} + \frac{(C_L - C_{L,mindrag})^2}{\pi e AR}$$

Here, e is Oswald efficiency factor and AR is Aspect Ratio of wing, obtained by dividing square of wing span and surface area.

The slope of the C_L vs C_D curve gives us C_L/C_D. From the equations of lift and drag, by cancelling the common terms, it can be seen that the ratio C_L/C_D is same as L/D.

The relationship between C_L and C_D is shown in figure below-

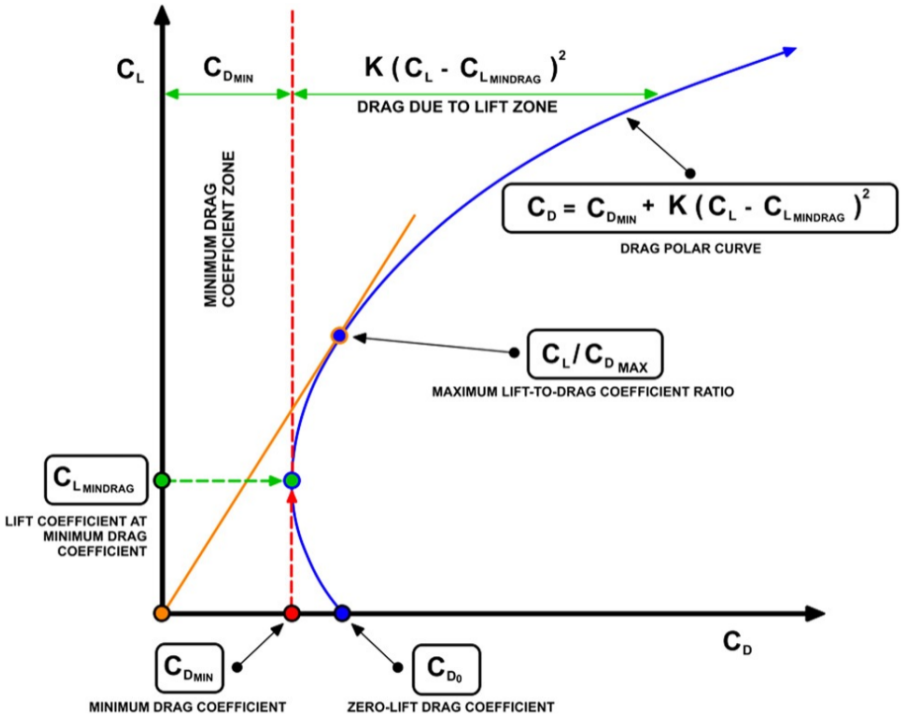


Figure 3: C_L vs C_D curve

The figure for C_L/C_D vs α is shown below-

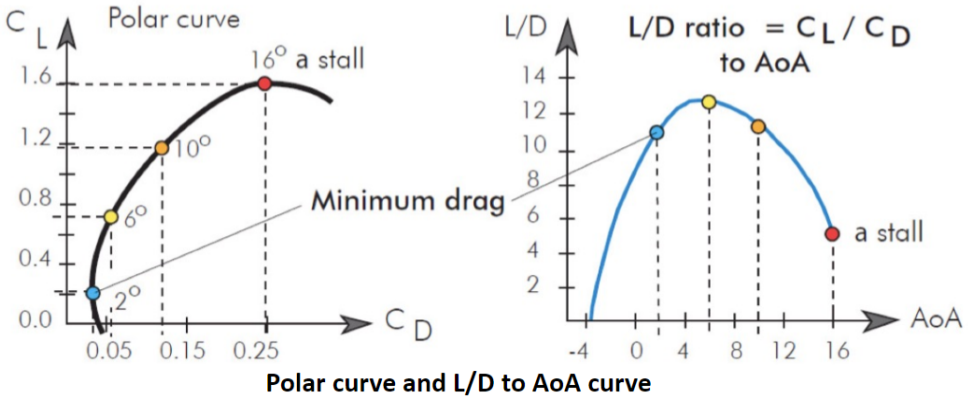


Figure 4: C_L/C_D vs α curve

1.3 Identification of random variables

Already having an idea from AE152 about various quantities as described above, we wanted to be able to see these for ourselves from data of real airfoils. So, our RVs were-

- **Angle of attack-** α
- **Lift coefficient-** C_L
- **Drag coefficient-** C_D
- **Ratio of lift and drag-** C_L/C_D

2 Data Collection

2.1 Problem

When we started our search for data, we could not find a uniform data set with α, C_L and C_D for multiple airfoils. Some databases which we found had different Reynolds no. for airfoils, some had different angles of attack etc.

2.2 Solution

To be able to streamline for data collection process, we used UIUC Airfoil Data Site to get *.dat* files which had coordinates of different airfoils. A sample image for *.dat* file of an airfoil 'HS1-430' whose coordinates were taken from the website is as follows-

```
HAM-STD HS1-430 AIRFOIL
     62.       62.

  0.00000   0.00000
  0.00050   0.01024
  0.00100   0.01456
  0.00200   0.02071
  0.00300   0.02547
  0.00500   0.03303
  0.00750   0.04057
  0.01000   0.04692
  0.02000   0.06627
  0.03000   0.08068
  0.04000   0.09241
  0.05000   0.10235
  0.06000   0.11097
  0.07000   0.11857
  0.08000   0.12534
  0.09000   0.13138
  0.10000   0.13680
  0.12000   0.14595
  0.14000   0.15325
  0.16000   0.15904
  0.18000   0.16517
```

Figure 5: HS1-430 airfoil coordinates .dat file

We downloaded data on 120 airfoils to represent our population. Next, we had to obtain data on α, C_L and C_D from these coordinate data files. We used open-source software XFOIL for this purpose.

In XFOIL, for streamlining entire analysis, we set our Reynolds No. as 50,000 and set α to vary from -10 to 40 degrees, increasing by 1 degree for each iteration. We did this procedure for each of our 120 airfoils and collected data in spreadsheets.

An image of Xfoil interface is as follows-

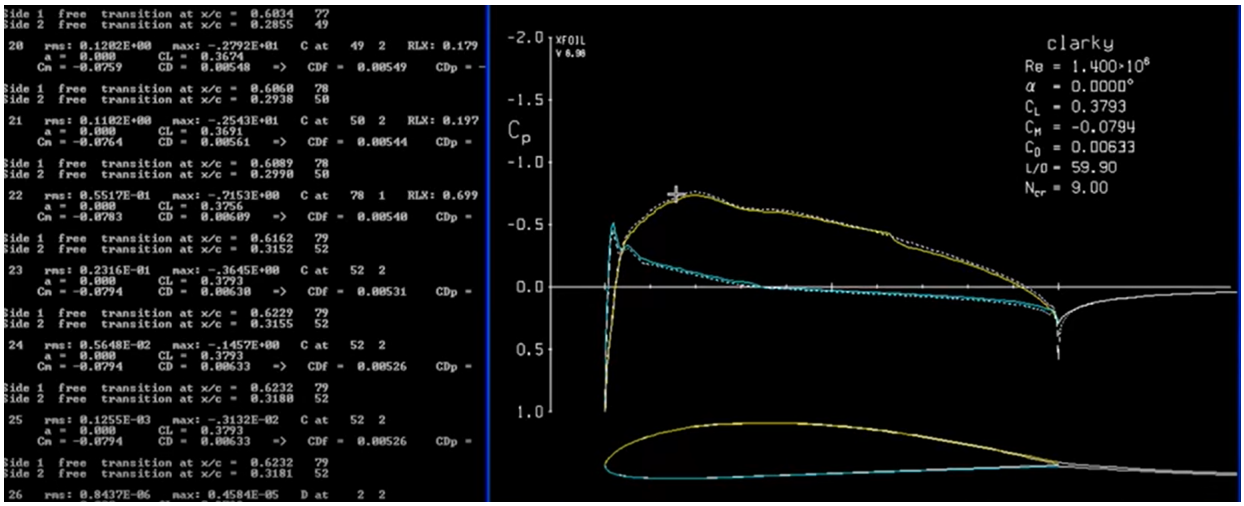


Figure 6: Interface of Xfoil used to obtain required data on airfoils

2.3 Data obtained from Xfoil

An image of data obtained from Xfoil after analysis of an airfoil is as follows-

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
2	-10	-0.4384	0.12881	0.12219	0.0191	1.0045	0.1895
3	-9	-0.4176	0.11613	0.10976	0.0148	1.0045	0.2164
4	-8	-0.4069	0.10404	0.09801	0.0106	1.0045	0.2522
5	-6	-0.3396	0.07879	0.07355	0.0144	1.0045	0.3839
6	-4	-0.1848	0.03621	0.02833	-0.0472	0.9397	0.1397
7	-3	-0.0158	0.02786	0.01841	-0.0546	0.8337	0.1847
8	-2	0.1519	0.02069	0.01389	-0.0524	0.7475	0.9955
9	-1	0.2378	0.02204	0.01375	-0.0481	0.682	0.9955
10	0	0.3271	0.02413	0.01482	-0.0439	0.6233	0.9955
11	1	0.4179	0.02713	0.01714	-0.0405	0.5684	0.9955
12	2	0.5086	0.03128	0.0209	-0.0384	0.5246	0.9955
13	3	0.5938	0.03703	0.02655	-0.0377	0.4921	0.9955
14	4	0.6548	0.04663	0.03636	-0.0395	0.4662	0.9955
15	5	0.6749	0.06109	0.051	-0.0438	0.4478	0.9955
16	7	0.5831	0.10122	0.09131	-0.061	0.5289	0.9955

TSAGI_R38 | s8066 | s8065 | s8064 | s4096 | s4095 | s4094

Figure 7: Data obtained from Xfoil

We pooled data from all our 120 airfoils to make 'Population'. We chose random 18 airfoils among these for 'Sample' analysis.

3 Population analysis

3.1 Dataset

Using data obtained from Xfoil, we pooled all data of 120 airfoils into 1 spreadsheet and made some modifications to get columns for α, C_L, C_D and C_L/C_D in spreadsheet. This served as our 'Population'.

	A	B	C	D
1	**Alpha**	**CL**	**CD**	**CL/CD**
2	-10	-0.2193	0.07184	-3.052616927
3	-9	-0.1955	0.06331	-3.087979782
4	-7	-0.4876	0.09399	-5.187785935
5	-6	-0.3128	0.0769	-4.067620286
6	-5	-0.4318	0.06286	-6.869233217
7	-4	-0.4131	0.04028	-10.25571003
8	-3	-0.329	0.033	-9.96969697
9	-2	-0.2311	0.02802	-8.247680228
10	-1	-0.1333	0.02405	-5.542619543
11	0	-0.0726	0.02172	-3.342541436
12	1	0.0029	0.02344	0.1237201365
13	2	0.1191	0.02643	4.506242906
14	3	0.2783	0.03092	9.000646831
15	4	0.5337	0.03039	17.56169793
16	5	0.6812	0.02116	32.19281664
17	6	0.7472	0.02922	25.57152635

Figure 8: Data set for population

3.2 Preliminary analysis

We begin by importing libraries and loading spreadsheet as pandas dataframe. By using df.describe(), we find true mean and true variance for all our random variables-

$\mu_{alpha} = 14.489120$, $\mu_{C_L} = 0.561269$, $\mu_{C_D} = 0.202071$, $\mu_{C_L/C_D} = 3.850024$

$\sigma_{alpha} = 14.460593$, $\sigma_{C_L} = 0.536662$, $\sigma_{C_D} = 0.147401$, $\sigma_{C_L/C_D} = 9.101799$

These were our parameters for true distribution.

The code snippets are in next page-

True population analysis

In [2]:
```
#Importing libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn
```

In [3]:
```
#Reading data file
df = pd.read_csv('True_population.csv')
df.head()
```

Out[3]:

	Alpha	CL	CD	CL/CD
0	-7	-0.5081	0.09509	-5.343359
1	-6	-0.4739	0.08063	-5.877465
2	-5	-0.4361	0.06743	-6.467448
3	-4	-0.1384	0.04052	-3.415597
4	-3	-0.2740	0.03935	-6.963151

In [4]:
```
df.describe()  #Shows mean, standard dev, median etc
```

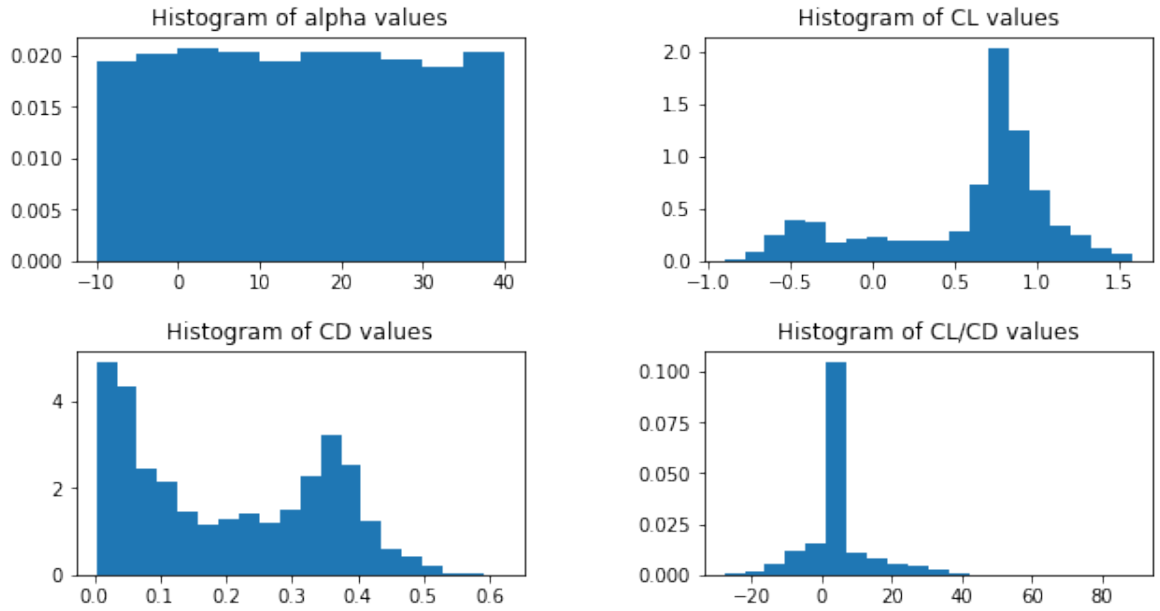
Out[4]:

	Alpha	CL	CD	CL/CD
count	5469.000000	5469.000000	5469.000000	5469.000000
mean	14.489120	0.561269	0.202071	3.850024
std	14.460593	0.536662	0.147401	9.101799
min	-10.000000	-0.907400	0.003050	-27.926728
25%	2.000000	0.267300	0.052850	1.860579
50%	14.000000	0.759300	0.181120	2.773353
75%	27.000000	0.886000	0.345860	4.541381
max	40.000000	1.579000	0.621980	88.470032

3.3 Histograms

First, histograms were made for each of our chosen random variables.

In [30]:
```
plt.figure(figsize=(10,5))
plt.subplot(2,2,1)
plt.hist(df['Alpha'],bins=10,density=True);   #Histogram for Alpha values
plt.title('Histogram of alpha values');       #It is almost a constant distribution, as expected
plt.subplot(2,2,2)                            #Histogram for C_L values
plt.hist(df['CL'],bins=20,density=True);
plt.title('Histogram of CL values');
plt.subplot(2,2,3)
plt.hist(df['CD'],bins=20,density=True);      #Histogram for C_D values
plt.title('Histogram of CD values');
plt.subplot(2,2,4)
plt.hist(df['CL/CD'],bins=20,density=True);   #Histogram for C_L/C_D values
plt.title('Histogram of CL/CD values');

# set the spacing between subplots
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
```

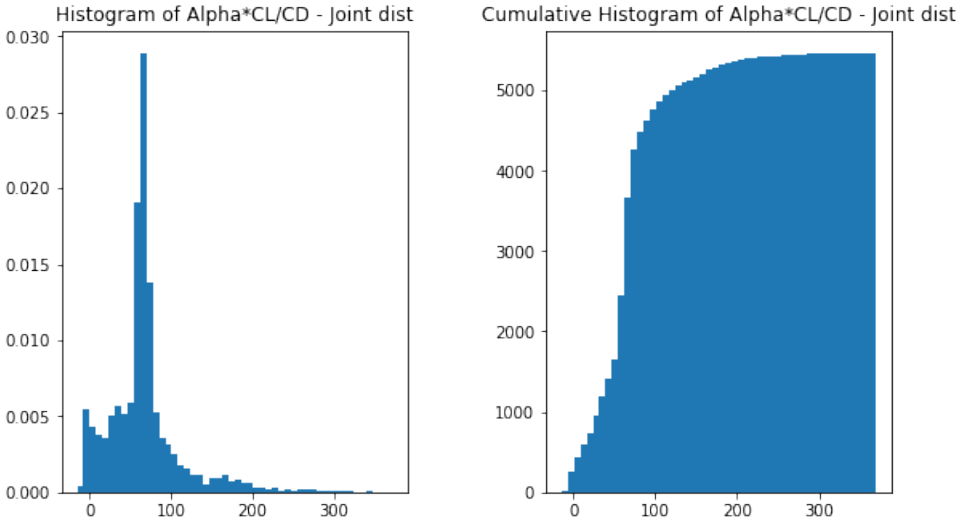


Since we had set α to vary from -10 to 40 degrees for each airfoil in X-foil, we get an almost uniform distribution in α histogram. In other histograms, it is seen that the data tends to center around some specific values; such as in range 0.5 - 1 for C_L and between 0.0 - 0.1, 0.3 - 0.4 for C_D.
These are the true distributions for our random variables.

3.4 Joint distributions

Next, we set out to make joint distributions. However, there is an issue here for our choice of random variables. Had our random variables been independent, we could have easily multiplied the CDFs to compute the joint distribution. But, in our case, we already know from theory that the pairs α, C_L and C_L, C_D are related. So, we only consider the joint distribution of α and C_L/C_D. As this pair of RVs is independent, their joint distribution PDF is same as multiplication of individual PDFs as implemented in code below-

```
In [38]: plt.figure(figsize=(8,5))
         plt.subplot(1,2,1)
         data = df['Alpha']*df['CL/CD']
         plt.hist(data, bins=50, density=True);  #Alpha and CL/CD
         plt.title('Histogram of Alpha*CL/CD - Joint dist');
         plt.subplot(1,2,2)
         data = df['Alpha']*df['CL/CD']
         plt.hist(data, bins=50, cumulative=True);  #Alpha and CL/CD
         plt.title('Cumulative Histogram of Alpha*CL/CD - Joint dist');

         # set the spacing between subplots
         plt.subplots_adjust(left=0,
                             bottom=0.1,
                             right=0.9,
                             top=0.9,
                             wspace=0.4,
                             hspace=0.4)
```



3.5 Correlations and inferences

Correlation coefficients were found out for pairs (α,C_L),(C_L,C_D) and (α,C_L/C_D).

Correlation coefficients

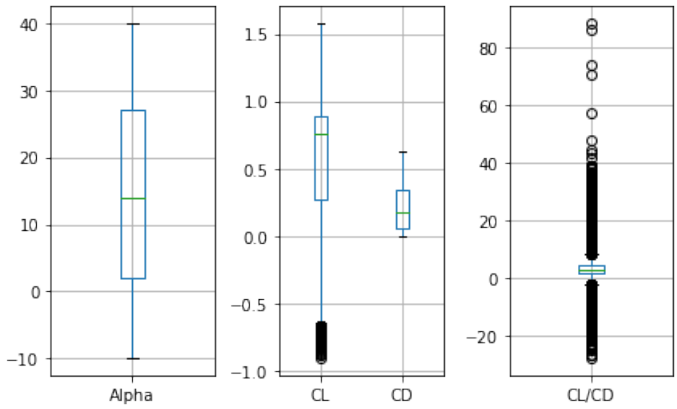
```
In [16]: df['Alpha'].corr(df['CL'])    #Alpha and CL correlation, since it is a positive expectedly linear relationship
Out[16]: 0.7639944808279484

In [17]: df['CL'].corr(df['CD'])      #CL and CD correlation, expected quadratic relationship
Out[17]: 0.5925022541536957

In [18]: df['Alpha'].corr(df['CL/CD'])          #Alpha and CL/CD correlation, no clear equation
Out[18]: 0.0852194692118693
```

We know that before stall starts, α and C_L are linearly correlated. In this light, since df also contains data points for stalling, correlation coefficient of 0.76 is understandable.
Since C_L and C_D have a quadratic relation and not linear, their correlation of 0.59 makes sense.
Also, α and C_L/C_D do not have any particular relationship (only a particular trend), hence a very low correlation coefficient.

3.6 Visualization of population data

We start by making box plots of α, C_L and C_D.

```
In [41]: plt.subplot(1,3,1)
         df.boxplot(['Alpha']);             #Alpha values
         plt.subplot(1,3,2)
         df.boxplot(['CL','CD']);          #CL and CD values
         plt.subplot(1,3,3)
         df.boxplot(['CL/CD']);         #CL/CD values, this seems to have lot of outliers

         # set the spacing between subplots
         plt.subplots_adjust(left=0,
                             bottom=0.1,
                             right=0.9,
                             top=0.9,
                             wspace=0.4,
                             hspace=0.4)
```

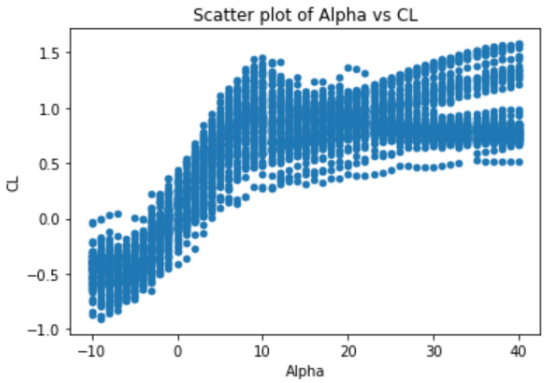


From third box plot, it can be seen that C_L/C_D has lot of outliers in data. The ranges of our random variables seen to be very different, hence the need of three different box plots here.

Now, we see the scatter plots of (α,C_L), (C_L,C_D) and (α,C_L/C_D).

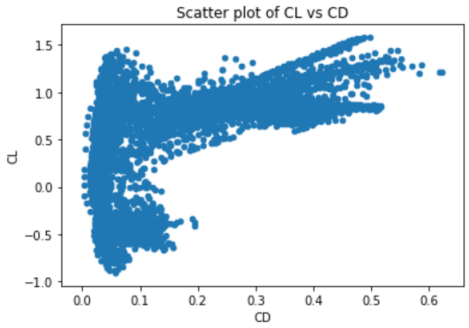
Scatter plots

```
In [8]: df.plot.scatter(x='Alpha',y='CL');
        plt.title('Scatter plot of Alpha vs CL');
```



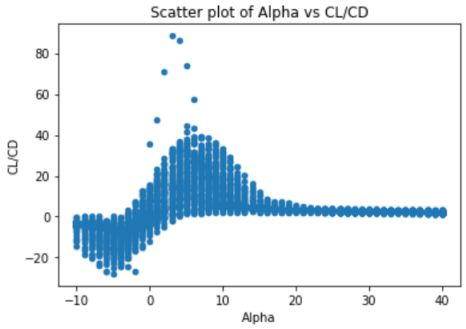
Since our population contains data of multiple airfoils, there seem to be multiple characteristic C_L vs α curves as discussed previously. So, the real data seems to follow empirical linear relationship of C_L and α before stall starts. In general, it can be said that stalling of airfoils in population happens around 10 degrees as visible in the scatter plot.

```
In [9]: df.plot.scatter(x='CD',y='CL');
        plt.title('Scatter plot of CL vs CD');
```



Since our population contains data of multiple airfoils, there seem to be multiple characteristic C_L vs C_D curves as discussed previously. So, the real data seems to follow the constant relationship of C_L and C_D for an infinite wing.

```
In [10]: df.plot.scatter(x='Alpha',y='CL/CD');
         plt.title('Scatter plot of Alpha vs CL/CD');
```



Most of the charecteristic curves seem to be following a particular trend of attaining minima around -10 to 0 degrees and maxima in range of 0 to 10 degrees. There seems to be some outliers too in scatter plot.

4 Sample analysis

4.1 Data set

We chose 18 airfoils at random from our population of 120 airfoils to make our sample. For each of these airfoils, we found mean of α, C_L and C_D. Code snippet for one of these airfoils is as shown below-

Sample analysis

We know true_mean and true_variance from true population analysis. We choose sample airfoils and find sample_mean.

In [22]:
```
#Choosing 3 random airfoils
for i in range(3):
    print(np.random.randint(low=1,high=21))
```

```
19
6
9
```

In [23]:
```
#Downloaded corresponding .csv file from spreadsheet and named sample1, sample2, sample3
#Airfoil 1
df1 = pd.read_csv('sample1.csv')
df1.head()
```

Out[23]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-7	-0.4341	0.08037	0.07274	-0.0048	1.0	0.2736
1	-6	-0.4152	0.06708	0.05981	0.0001	1.0	0.3455
2	-5	-0.3913	0.05610	0.04912	0.0049	1.0	0.4291
3	-4	-0.2338	0.03651	0.02677	-0.0308	1.0	0.1432
4	-3	-0.1358	0.02724	0.01630	-0.0310	1.0	0.1297

In [24]:
```
df1.describe()
```

Out[24]:

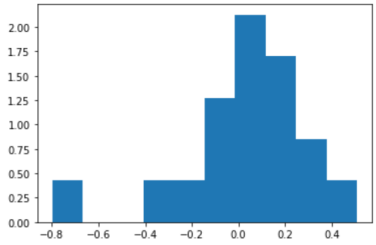
	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000
mean	15.717391	0.621985	0.213179	0.204698	-0.072200	0.310687	0.901200
std	13.772215	0.402052	0.148769	0.150265	0.038739	0.398112	0.261374
min	-7.000000	-0.434100	0.018290	0.008370	-0.138600	0.012200	0.129700
25%	4.250000	0.733150	0.051397	0.041103	-0.100375	0.039675	1.000000
50%	15.500000	0.764950	0.253180	0.244430	-0.078000	0.091900	1.000000
75%	26.750000	0.859125	0.346440	0.337975	-0.033975	0.624900	1.000000
max	40.000000	0.973100	0.420960	0.416060	0.004900	1.000000	1.000000

As seen above, x_bar_alpha = 15.717391, x_bar_CL = 0.621985, x_bar_CD = 0.213179 and N = 46

4.2 Sampling distribution for α

We computed Z by using α_{mean} obtained for each of the 18 airfoils and true mean and true variance from population analysis. $Z_\alpha = (\alpha_{mean} - \mu)/(\sigma/n^{1/2})$

```
In [5]: #Sampling distribution for alpha
Alpha_sample_means = np.array([15.717391,16.222222,15.0,14.836735,15.122449,15.040816,13.50,14.60,14.142857,14.720,11.769231,14.2
true_mean_alpha = 14.489120   #From population analysis
true_std_alpha = 14.460593    #From population analysis
n = 18     #Sample size
Z_alpha = np.sqrt(n)*(Alpha_sample_means - true_mean_alpha)/true_std_alpha
plt.hist(Z_alpha, density=True);
```



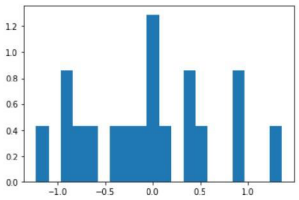
Next, we checked whether mean of our sampling distribution is close to true mean as predicted by Central Limit Theorem. Also, we computed 95% confidence interval for true mean of α as shown below-

```
In [6]: #CLT mean verification
print('Mean of sample is',np.mean(Alpha_sample_means),'while true mean is',true_mean_alpha,'. These are almost same in accordance

Mean of sample is 14.610455611111112 while true mean is 14.48912 . These are almost same in accordance with CLT.

In [7]: #95% CI for true mean
a = np.mean(Alpha_sample_means) - 1.96*true_std_alpha/np.sqrt(n)
b = np.mean(Alpha_sample_means) + 1.96*true_std_alpha/np.sqrt(n)
print('95% CI for true mean is given by interval [',a,b,']')

95% CI for true mean is given by interval [ 7.930002475862348 21.290908746359875 ]
```

4.3 Sampling distribution for C_L

We computed Z by using C_{Lmean} obtained for each of the 18 airfoils and true mean and true variance from population analysis. $Z_{C_L} = (C_{Lmean} - \mu)/(\sigma/n^{1/2})$

```
In [20]: #sampling distribution for CL
CL_sample_mean=sm['x_bar_CL']
true_mean_CL=0.561269 #from population analysis
true_std_CL=0.536662 #from population analysis
n=18 #sample size
Z_CL=np.sqrt(n)*(CL_sample_mean-true_mean_CL)/true_std_CL
plt.hist(Z_CL,density=True, bins=20);
```



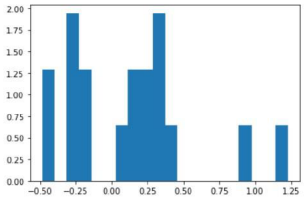
Next, we checked whether mean of our sampling distribution is close to true mean as predicted by Central Limit Theorem. Also, we computed 95% confidence interval for true mean of C_L as shown below-

```
In [19]: #CLT mean verfication
         print('Mean of sample is', np.mean(CL_sample_mean),'while true mean is',true_mean_CL,
               'This are almost same in accordance with CLT')

         Mean of sample is 0.5577212777777779 while true mean is 0.561269 This are almost same in accordance with CLT

In [25]: #95% CI for true mean
         c=np.mean(CL_sample_mean)-1.96*true_std_CL/np.sqrt(n)
         d=np.mean(CL_sample_mean)+1.96*true_std_CL/np.sqrt(n)
         print('95% CI for true mean is given by interval[',c,d,']' )

         95% CI for true mean is given by interval[ 0.3097960826997564 0.8056464728557995 ]
```

4.4 Sampling distribution for C_D

We computed Z by using C_{Dmean} obtained for each of the 18 airfoils and true mean and true variance from population analysis. $Z_{C_D} = (C_{Dmean} - \mu)/(\sigma/n^{1/2})$

```
In [17]: #sampling ditribution for CD
         CD_sample_mean=sm['x_bar_CD']
         true_mean_CD=0.202071
         true_std_CD=0.147401
         n=18
         z_CD=np.sqrt(n)*(CD_sample_mean-true_mean_CD)/true_std_CD
         plt.hist(z_CD,density=True,bins=20);
```



Next, we checked whether mean of our sampling distribution is close to true mean as predicted by Central Limit Theorem. Also, we computed 95% confidence interval for true mean of C_D as shown below-

```
In [18]: #CLT mean verfication
         print('Mean of sample is', np.mean(CD_sample_mean),'while true mean is',true_mean_CD,
               'This are almost same in accordance with CLT')

         Mean of sample is 0.2065297222222222 while true mean is 0.202071

In [27]: #95% CI for true mean
         e=np.mean(CD_sample_mean)-1.96*true_std_CD/np.sqrt(n)
         f=np.mean(CD_sample_mean)+1.96*true_std_CD/np.sqrt(n)
         print('95% CI for true mean is given by interval[',e,f,']' )

         95% CI for true mean is given by interval[ 0.13843393440848573 0.2746255100359587 ]
```

4.5 Alternate approach- Error bars

Error bars are graphical representations of the variability of data and used on graphs to indicate the error or uncertainty in a reported measurement. They give a general idea of how precise a measurement is, or conversely, how far from the reported value the true (error free) value might be. Error bars often represent one standard deviation of uncertainty, one standard error, or a particular confidence interval (e.g., a 95% interval).

Here, we have used a red line to show true mean and the blue dot to show sample mean of each of chosen 18 airfoils. The ends of blue line indicate range of 95% confidence interval.

In [6]:
```
#read sample file data
sm=pd.read_csv('confidence interval.csv')
```

In [4]:
```
sm['Error_CL']=1.96*(sm['std_CL']/sm['N']**0.5)
sm['CI_low_CL']=sm['x_bar_CL']-sm['Error_CL']
sm['CI_high_CL']=sm['x_bar_CL']+sm['Error_CL']
CI=(sm['CI_low_CL'],sm['CI_high_CL'])
yy=(sm['CI_high_CL']-sm['CI_low_CL'])/2
```

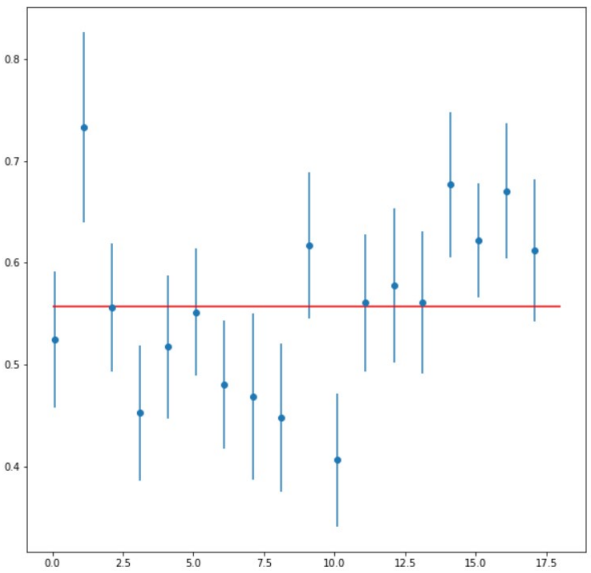
In [5]:
```
sm #data
```

Out[5]:

	sample no	x_bar_CL	std_CL	x_bar_CD	std_CD	N	Error_CL	CI_low_CL	CI_high_CL
0	sample1	0.525102	0.485898	0.206425	0.156059	48	0.137461	0.387641	0.662563
1	sample2	0.733065	0.665441	0.211494	0.157875	46	0.192303	0.540762	0.925368
2	sample3	0.555987	0.448153	0.203343	0.159290	46	0.129510	0.426477	0.685497
3	sample4	0.452924	0.492914	0.191322	0.139995	50	0.136629	0.316295	0.589553
4	sample5	0.517879	0.505721	0.209210	0.148338	47	0.144583	0.373296	0.662462
5	sample6	0.551827	0.451872	0.234300	0.123359	48	0.127835	0.423992	0.679662
6	sample7	0.480272	0.467629	0.194131	0.143351	50	0.129620	0.350652	0.609892
7	sample8	0.469213	0.535213	0.185255	0.130048	39	0.167977	0.301236	0.637190
8	sample9	0.448179	0.529597	0.188863	0.146092	48	0.149824	0.298355	0.598003
9	sample10	0.616994	0.523883	0.196311	0.142964	48	0.148207	0.468787	0.765201
10	sample11	0.406650	0.486778	0.193279	0.139749	50	0.134928	0.271722	0.541578
11	sample12	0.560765	0.493112	0.196405	0.143213	49	0.138071	0.422694	0.698836
12	sample13	0.577718	0.556172	0.244752	0.184363	49	0.155728	0.421990	0.733446
13	sample14	0.560786	0.514026	0.213666	0.162598	49	0.143927	0.416859	0.704713
14	sample15	0.676559	0.527508	0.207851	0.165082	49	0.147702	0.528857	0.824261
15	sample16	0.621985	0.402052	0.213179	0.148769	46	0.116187	0.505798	0.738172
16	sample17	0.670533	0.467451	0.216851	0.141838	45	0.136580	0.533953	0.807113
17	sample18	0.612545	0.525075	0.212898	0.147792	51	0.144109	0.468436	0.756654

In [62]:
```
#ploting error graph for each sample
plt.figure(figsize=(10,10))
plt.errorbar(x=np.arange(0.1,18,1),
             y=sm['x_bar_CL'],
             yerr=yy,
             fmt='o')
plt.hlines(xmin=0, xmax=18,y=sm['x_bar_CL'].mean(),color='r')
```

Out[62]:
```
<matplotlib.collections.LineCollection at 0x1eb3d6c44f0>
```



In [7]:
```
sm['Error_CD']=1.96*(sm['std_CD']/sm['N']**0.5)
sm['CI_low_CD']=sm['x_bar_CD']-sm['Error_CD']
sm['CI_high_CD']=sm['x_bar_CD']+sm['Error_CD']
CI=(sm['CI_low_CD'],sm['CI_high_CD'])
yR=(sm['CI_high_CD']-sm['CI_low_CD'])/2
```

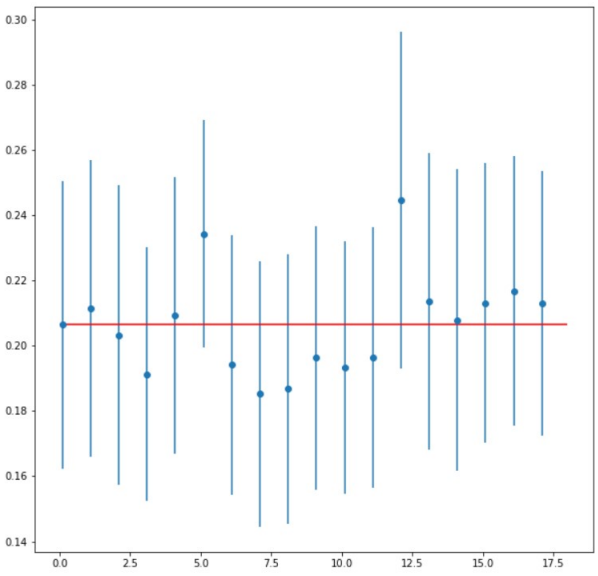
In [8]:
```
sm
```

Out[8]:

	sample no	x_bar_CL	std_CL	x_bar_CD	std_CD	N	Error_CD	CI_low_CD	CI_high_CD
0	sample1	0.525102	0.485898	0.206425	0.156059	48	0.044149	0.162276	0.250574
1	sample2	0.733065	0.665441	0.211494	0.157875	46	0.045624	0.165870	0.257118
2	sample3	0.555987	0.448153	0.203343	0.159290	46	0.046033	0.157310	0.249376
3	sample4	0.452924	0.492914	0.191322	0.139995	50	0.038805	0.152517	0.230127
4	sample5	0.517879	0.505721	0.209210	0.148338	47	0.042409	0.166801	0.251619
5	sample6	0.551827	0.451872	0.234300	0.123359	48	0.034898	0.199402	0.269198
6	sample7	0.480272	0.467629	0.194131	0.143351	50	0.039735	0.154396	0.233866
7	sample8	0.469213	0.535213	0.185255	0.130048	39	0.040816	0.144439	0.226071
8	sample9	0.448179	0.529597	0.186863	0.146092	48	0.041330	0.145533	0.228193
9	sample10	0.616994	0.523883	0.196311	0.142964	48	0.040445	0.155866	0.236756
10	sample11	0.406650	0.486778	0.193279	0.139749	50	0.038736	0.154543	0.232015
11	sample12	0.560765	0.493112	0.196405	0.143213	49	0.040100	0.156305	0.236505
12	sample13	0.577718	0.556172	0.244752	0.184363	49	0.051622	0.193130	0.296374
13	sample14	0.560786	0.514026	0.213666	0.162598	49	0.045527	0.168139	0.259193
14	sample15	0.676559	0.527508	0.207851	0.165082	49	0.046223	0.161628	0.254074
15	sample16	0.621985	0.402052	0.213179	0.148769	46	0.042992	0.170187	0.256171
16	sample17	0.670533	0.467451	0.216851	0.141838	45	0.041442	0.175409	0.258293
17	sample18	0.612545	0.525075	0.212898	0.147792	51	0.040562	0.172336	0.253460

In [22]:
```
#ploting error graph for each sample
plt.figure(figsize=(10,10))
plt.errorbar(x=np.arange(0.1,18,1),
             y=sm['x_bar_CD'],
             yerr=yR,
             fmt='o')
plt.hlines(xmin=0, xmax=18,y=sm['x_bar_CD'].mean(),color='r')
```

Out[22]: <matplotlib.collections.LineCollection at 0x1eb59c48940>



5 Linear Regression of α and C_L

5.1 Theory and Expectations

Simple linear regression is an approach for predicting a response using a single feature.
It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).
Here Variable(x) is α and Variable(y) is C_L
The lift coefficient is always defined as the aerodynamic lift divided by the dynamic pressure and some reference area

5.2 Code

In order to perform linear regression between α and C_L we made three functions in python to get the output of their regression :

1. Estimate coef(x, y) : Here we we counted the number of observations and perform linear Regression using the formulae for it.

2. Plot regression line : Here we have done plotting the actual points as scatter plot, predicted response vector and plotted the regression line.

3. main() : Here we call the α and its corresponding C_L values.

So out of 120 populations, we have chosen any of the 2 airfoils to check whether is satisfies or not.
Note : For this we have not taken all values of α ranging - 10 to (35 to 40) and corresponding C_L values. Because after a certain α we get a stall.So for getting a linear relationship we have reduce the number of observations of α
.

In [6]:
```
# consider any Airfoil data (one at a time)

df1 = pd.read_csv('sample12.csv')
df1
```

In [19]:
```
# the random variables considered here are Alpha and Cl

Alpha = np.array(df1["alpha"][:20])
Cl = np. array(df1["CL"][:20])

print("Alpha values : ",Alpha)
print("-------------------------------------------------------------------------------
print("CL values :",Cl)
```

```
Alpha values :  [-10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   6   8
9
  10  11]
--------------------------------------------------------------------------------
------------------
CL values : [-0.5315 -0.7619 -0.7784 -0.7415 -0.668  -0.6055 -0.585  -0.5708 -0.5019
 -0.4454 -0.4132 -0.3615 -0.2706 -0.1274  0.0069  0.1822  0.3686  0.4608
  0.5839  0.5987]
```

In [25]:
```
#Regression

def estimate_coeff(x,y):
    n= np.size(x)                                   #number of observations/points

    m_x=np.mean(x)                                  #mean of x vector
    m_y=np.mean(y)                                  #mean of y vector

    SS_xy = np.sum(y*x) - n*m_y*m_x                      #cross deviation
    SS_xx = np.sum(x*x) - n*m_x*m_x                      #deviation about x

    b1=SS_xy / SS_xx
    b0=m_y - b1*m_x                              #regression coefficients

    return(b0,b1)

def plot_regression_line(x,y,b):
    plt.scatter(x,y,color='m',marker="o",s=30)

    y_predict =b[0] +b[1]*x                         #predicted response vector

    plt.plot(x,y_predict,color='g')                 #regression line
```

```
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()

def main():                                        #observations / data
    x=Alpha
    y=Cl

    b=estimate_coeff(x,y)                          #coefficient estimation
    print("Estimated coefficients:\nb_0 ={}  \nb_1 ={}".format(b[0],b[1]))

    plot_regression_line(x,y,b)

if __name__=="__main__":
    main()
```

```
Estimated coefficients:
b_0 =-0.25469502417528606
b_1 =0.06759951649427749
```

5.3 Observations

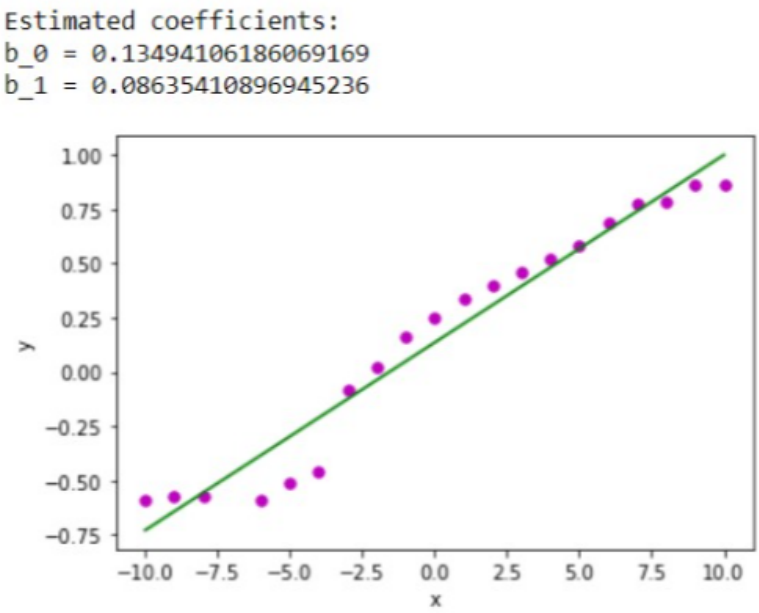


Figure 9: Sample Airfoil 1

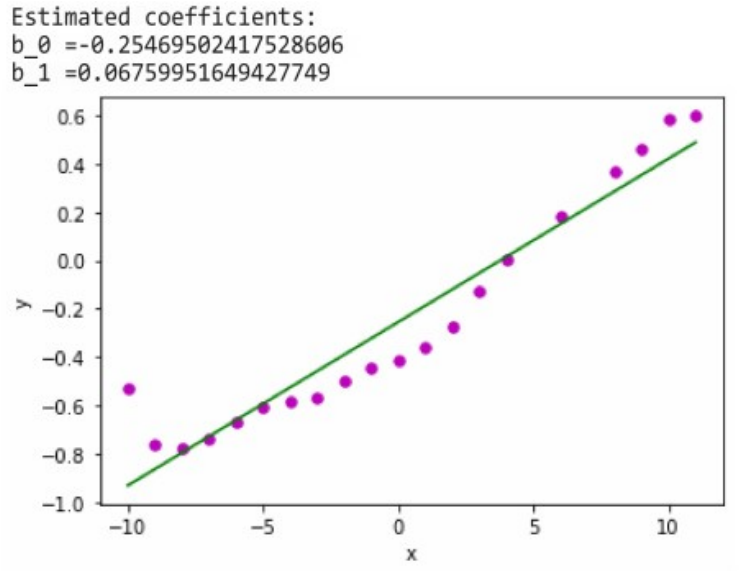


Figure 10: Sample Airfoil 2

From the above plots, it is clear that there is a linear relationship between α and C_L. This linear relationship is only valid till the airfoil remains in an unstalled state. After an airfoil stalls, C_L begins to fall off drastically with further increase in α.

6 Relation between C_L and C_D

6.1 Theory and Expectations

This subsection will see us exploring the relationship between C_L and C_D for an airfoil. The *Drag Polar* equation for a wing is given as,

$$C_D = C_{D,0} + \frac{C_L^2}{\pi e AR}$$

Where $C_{D,0}$ is the value of C_D when $C_L = 0$ In the limit as the size of the wing tends to ∞, or in other words, $AR \to \infty$, we can clearly see that this equation reduces to

$$C_D = C_{D,0}$$

This tells us that the value of C_D can be expected to be relatively constant with changes in C_L for an *unstalled* airfoil.
After an airfoil stalls, we see that the value of C_D increases even with very small changes in C_L. This follows from the fact that as we increase the angle of attack, the airflow over an airfoil begins to separate from the surface of the airfoil, causing the *pressure drag* to rise dramatically.
Before the experiment was performed, the expected relation was a relatively constant value of C_D followed by a sudden and dramatic increase in C_D for a relatively constant value of C_L.

6.2 Code

The code is based on using a sample of 6 randomly chosen airfoils, and independently checking for the expected observations in all 6 airfoils. For each of the airfoils, the steps followed are as folows:

1. Load the sample onto a dataframe using Pandas
2. Defining a function to return the C_L and C_D series of the airfoil dataframe as numpy arrays
3. Defining a function to determine the value of C_D when the value of C_L is zero, followed by plotting the scatter plot of C_L and C_D along with the line $y = C_{D,0}$.
4. Using all these function to generate plots for all airfoils

[113]:
```
##Loading up my sample airfoils
af_1 = pd.read_csv('sample1.csv')
af_2 = pd.read_csv('sample2.csv')
af_3 = pd.read_csv('sample3.csv')
```

[98]:
```
##Only for airfoil 1
af_1.head()
```

[98]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-10	-0.4640	0.11616	0.10841	-0.0202	1.0	0.2521
1	-9	-0.4423	0.10394	0.09636	-0.0150	1.0	0.3338
2	-8	-0.4393	0.09302	0.08567	-0.0104	1.0	0.3878
3	-7	-0.4522	0.08112	0.07407	-0.0044	1.0	0.4209
4	-6	-0.6211	0.05108	0.04202	-0.0189	1.0	0.1461

[111]:
```
##Regression
def return_array(x, n = 1): #Will return a series in the form of a np array, given the index
    if(n == 1):
        return af_1[x].to_numpy()
    if(n == 2):
        return af_2[x].to_numpy()
    if(n == 3):
        return af_3[x].to_numpy()
def plot_regression_line(x, y, n = 1):
    plt.scatter(x, y, color = 'm', marker = 'o', s = 30)
    if(n == 1):
        y_p = af_1['CD'][11]*np.ones_like(x) #Since CL = 0 usually around alpha = 0
    if(n == 2):
        y_p = af_2['CD'][11]*np.ones_like(x)
    if(n == 3):
        y_p = af_3['CD'][12]*np.ones_like(x)
    plt.plot(x, y_p, color = 'g')
```

[103]:
```
CL, CD = return_array('CL'), return_array('CD') #Returns np arrays of CL and CD
plot_regression_line(CL, CD)
```

[104]:
```
##Airfoil 2
af_2.head()
```

[104]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-10	-0.3883	0.14588	0.13794	-0.0280	1.0	0.1992
1	-9	-0.3487	0.11618	0.10840	-0.0215	1.0	0.2760
2	-8	-0.4174	0.12692	0.11964	-0.0107	1.0	0.2984
3	-7	-0.4026	0.09807	0.09103	-0.0075	1.0	0.3242
4	-6	-0.3988	0.08941	0.08264	0.0024	1.0	0.3612

[105]:
```
CL, CD = return_array('CL', 2), return_array('CD', 2) #Returns np arrays of CL and CD
plot_regression_line(CL, CD, 2)
```

[116]:
```
##Airfoil 3
af_3.head()
```

[116]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-10	-0.5023	0.12573	0.11833	-0.0237	1.0	0.2552
1	-9	-0.5375	0.12332	0.11625	-0.0177	1.0	0.2997
2	-7	-0.5756	0.07985	0.07317	-0.0193	1.0	0.2347
3	-6	-0.6331	0.05105	0.04257	-0.0249	1.0	0.1358
4	-5	-0.5689	0.03848	0.02813	-0.0201	1.0	0.1383

[117]:
```
CL, CD = return_array('CL', 3), return_array('CD', 3) #Returns np arrays of CL␣
↪and CD
plot_regression_line(CL, CD, 3)
```

6.3 Observations

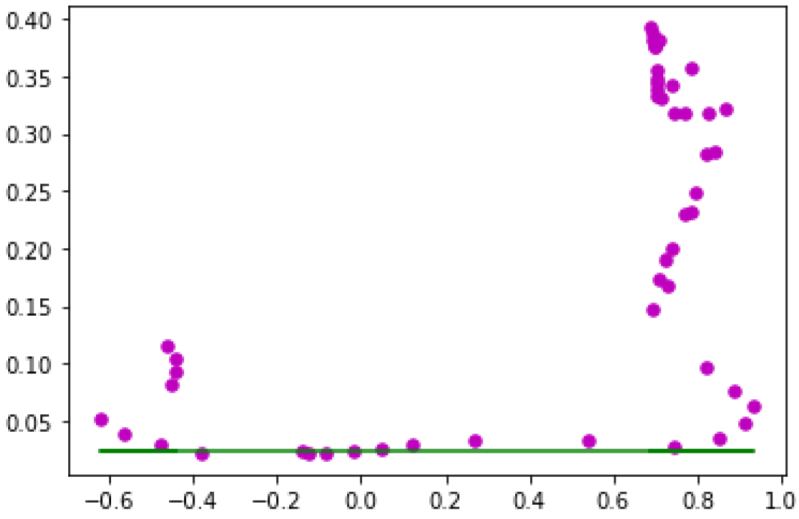


Figure 11: Sample Airfoil 1

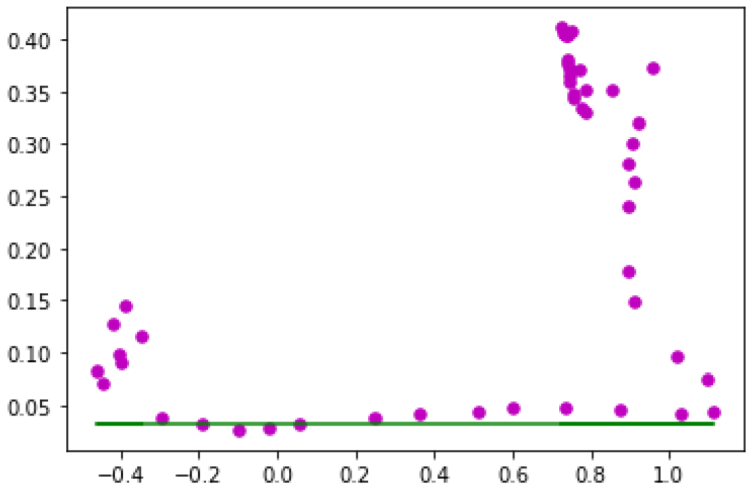


Figure 12: Sample Airfoil 2

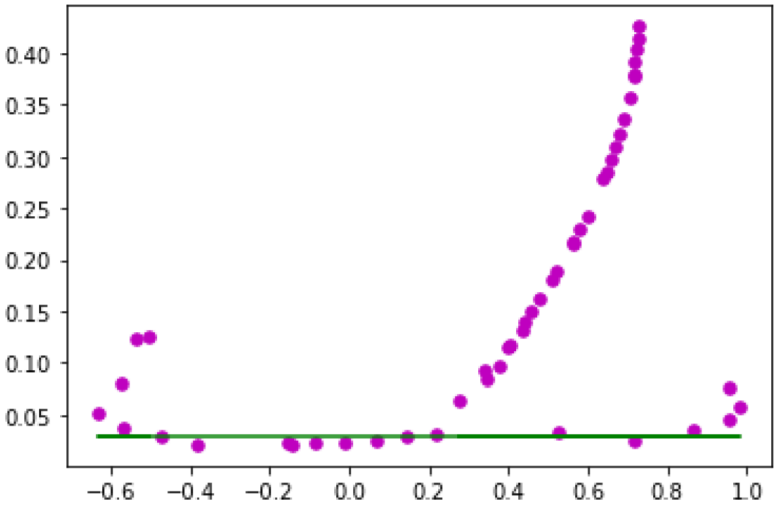


Figure 13: Sample Airfoil 3

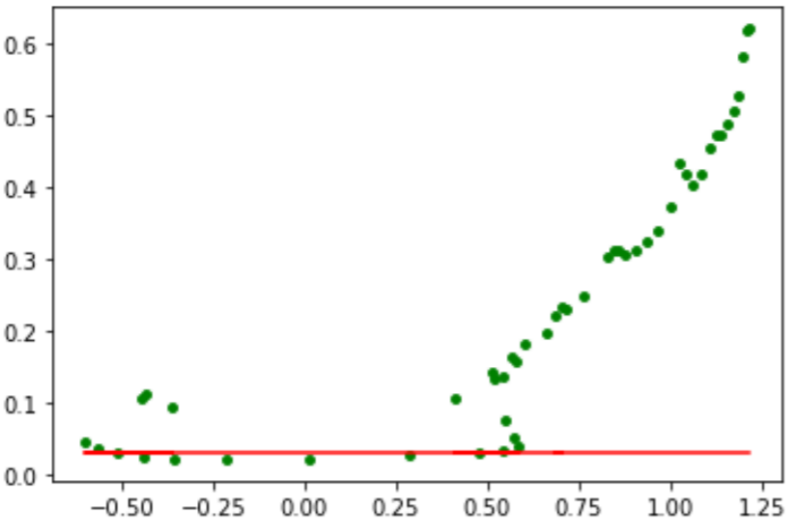


Figure 14: Sample Airfoil 4

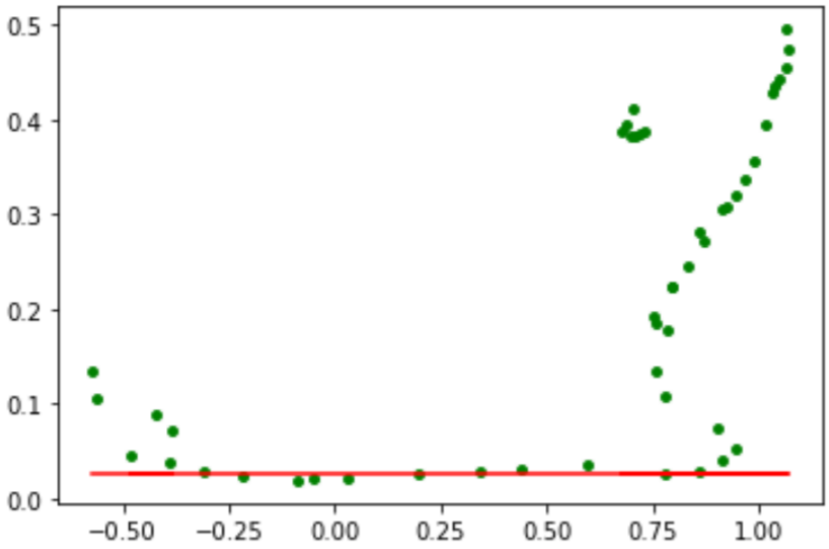


Figure 15: Sample Airfoil 5

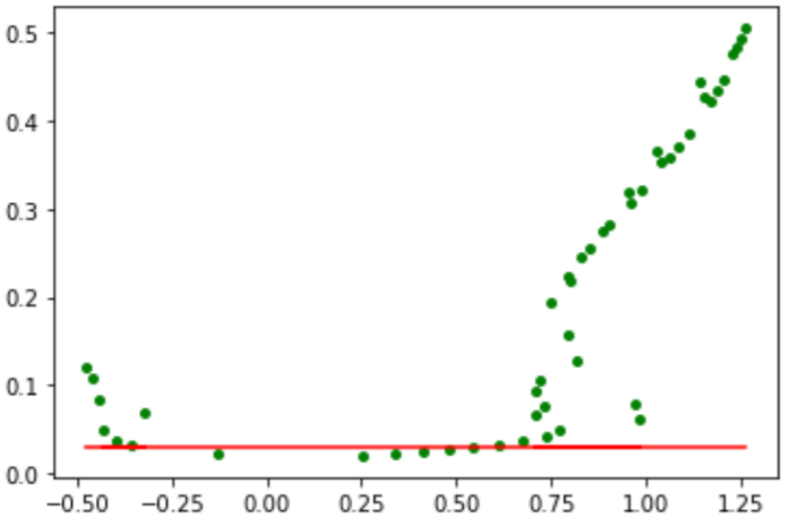


Figure 16: Sample Airfoil 6

From the observations, one can clearly see the following :

1. There is a set of values of C_L for which the value of C_D remains constant. This corresponds to the values obtained for an unstalled airfoil.

2. There is a set of value of C_L for which there are multiple values of C_D possible. This corresponds to the values obtained for a stalled airfoil.

3. There also seems to be a few negative values of C_L for which C_D changes dramatically. This corresponds to the values observed for an inverted stalled airfoil.

7 Hypothesis testing

7.1 Hypothesis Formulation

We state our hypothesis as follows:

$$H_0 : \text{Lift Slope of all airfoils is } 2\pi \text{ per unit radians, or } 2\pi^2/180 \text{ per unit } ^\circ$$

$$H_1 : \text{Lift slope of all airfoils is not } 2\pi \text{ per unit radians, or } 2\pi^2/180 \text{ per unit } ^\circ$$

7.2 Background

The above hypothesis follows from the *Thin-Airfoil theory*.
The *Thin-Airfoil theory* is a simple theory of airfoils that relates angle of attack to lift for incompressible, inviscid flows. It was devised by German mathematician Max Munk and further refined by British aerodynamicist Hermann Glauert and others in the 1920s. The theory idealizes the flow around an airfoil as two-dimensional flow around a thin airfoil. It can be imagined as addressing an airfoil of zero thickness and infinite wingspan.

The theory has a number of consequences regarding important properties of airfoils in 2-D flows:

1. For a symmetric airfoil,centre of pressure and the aerodynamic centre are coincident and both lie exactly a quarter of the chord behind the leading edge.
2. For a cambered airfoil, Aerodynamic centre lies exactly one quarter chord behind the leading edge.
3. The lift slope of an airfoil is 2π per unit radians.

7.3 Code

The code is based on using a sample of 3 randomly chosen airfoils, and using a t-distribution to calculate the critical region for our chosen level of significance (5%). To test our hypothesis, we have made an important assumption that the lift coefficients of airfoils are distributed normally and centred at $\mu = \dfrac{2\pi^2}{180}$. The steps followed were as follows :

1. Load the sample onto a dataframe using Pandas
2. Calculating the lift slope of each of the sample airfoils using linear regression
3. Calculating the sample mean S and taking the value of μ to be $\dfrac{2\pi^2}{180}$
4. Calculating the random variable $T = \dfrac{\bar{X} - \mu}{S/\sqrt{N}}$

5. Calculating the value of critical region for level of significance 5%

6. Using plots to see if the calculated T-value lies inside the critical region or not

In [2]:
```
af_1 = pd.read_csv('sample1.csv')
af_2 = pd.read_csv('sample2.csv')
af_3 = pd.read_csv('sample3.csv')
```

In [22]:
```
def return_series(x, n = 1):
    if n == 1:
        return af_1[x][:20].to_numpy()
    if n == 2:
        return af_2[x][:20].to_numpy()
    if n == 3:
        return af_3[x][:20].to_numpy()
def fit(x, y):
    return np.polyfit(x, y, 1)
def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = 'm', marker = 'o', s = 30)
    y_p = b[0]*x + b[1]
    plt.plot(x, y_p, color = 'g')
```

In [23]:
```
af_1.head()
```

Out[23]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-10	-0.4640	0.11616	0.10841	-0.0202	1.0	0.2521
1	-9	-0.4423	0.10394	0.09636	-0.0150	1.0	0.3338
2	-8	-0.4393	0.09302	0.08567	-0.0104	1.0	0.3878
3	-7	-0.4522	0.08112	0.07407	-0.0044	1.0	0.4209
4	-6	-0.6211	0.05108	0.04202	-0.0189	1.0	0.1461

In [24]:
```
CL, alpha = return_series('CL'), return_series('alpha')
B_1 = fit(alpha, CL)
plot_regression_line(alpha, CL, B_1)
print(B_1)
```

```
[0.08381872 0.05245436]
```

In [11]:
```
af_2.head()
```

Out[11]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-10	-0.3883	0.14588	0.13794	-0.0280	1.0	0.1992
1	-9	-0.3487	0.11618	0.10840	-0.0215	1.0	0.2760
2	-8	-0.4174	0.12692	0.11964	-0.0107	1.0	0.2984
3	-7	-0.4026	0.09807	0.09103	-0.0075	1.0	0.3242
4	-6	-0.3988	0.08941	0.08264	0.0024	1.0	0.3612

In [25]:
```
CL, alpha = return_series('CL', n = 2), return_series('alpha', n = 2)
B_2 = fit(alpha, CL)
plot_regression_line(alpha, CL, B_2)
print(B_2)
```

```
[0.08651075 0.14668038]
```

In [14]:
```
af_3.head()
```

Out[14]:

	alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
0	-10	-0.5023	0.12573	0.11833	-0.0237	1.0	0.2552
1	-9	-0.5375	0.12332	0.11625	-0.0177	1.0	0.2997
2	-7	-0.5756	0.07985	0.07317	-0.0193	1.0	0.2347
3	-6	-0.6331	0.05105	0.04257	-0.0249	1.0	0.1358
4	-5	-0.5689	0.03848	0.02813	-0.0201	1.0	0.1383

In [26]:
```
CL, alpha = return_series('CL', n = 3), return_series('alpha', n = 3)
B_3 = fit(alpha, CL)
plot_regression_line(alpha, CL, B_3)
print(B_3)
```

```
[0.09100176 0.03171929]
```

In [45]:
```
lift_slope = np.array([B_1, B_2, B_3])
n = 3
X_b = np.mean(lift_slope)
S = np.std(lift_slope, ddof = 1)
trv = t(n - 1)
mu = 2*np.pi*np.pi/180
T = (X_b - mu)/(S/np.sqrt(n))
los = 0.025
cmin, cmax = trv.ppf([los, 1 - los])
x = np.linspace(-5, 5)
plt.plot(x, trv.pdf(x));
plt.vlines([cmin, cmax], 0.00, ymax = 0.2, color = 'r');
plt.vlines([T], 0.00, ymax = 0.2, color = 'm');
plt.legend(['t-distribution', 'c_min, c_max', 'T-value']);
```

7.4 Result

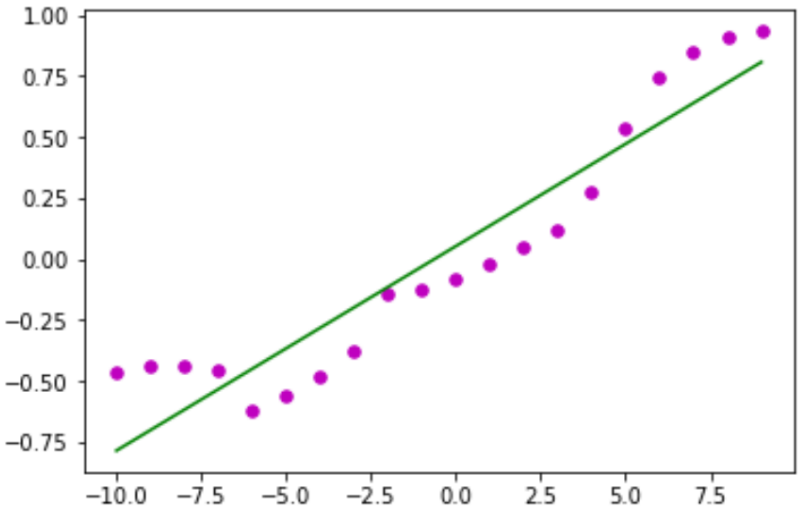


Figure 17: Sample Airfoil 1

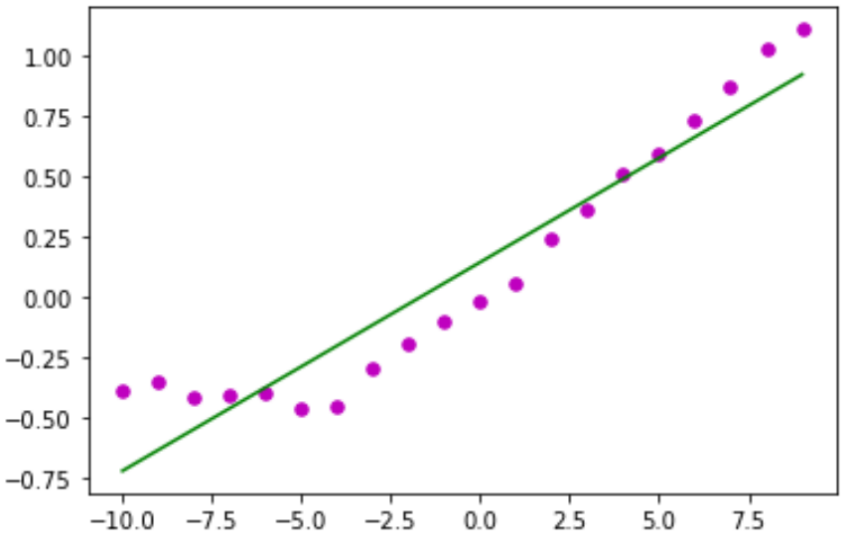


Figure 18: Sample Airfoil 2

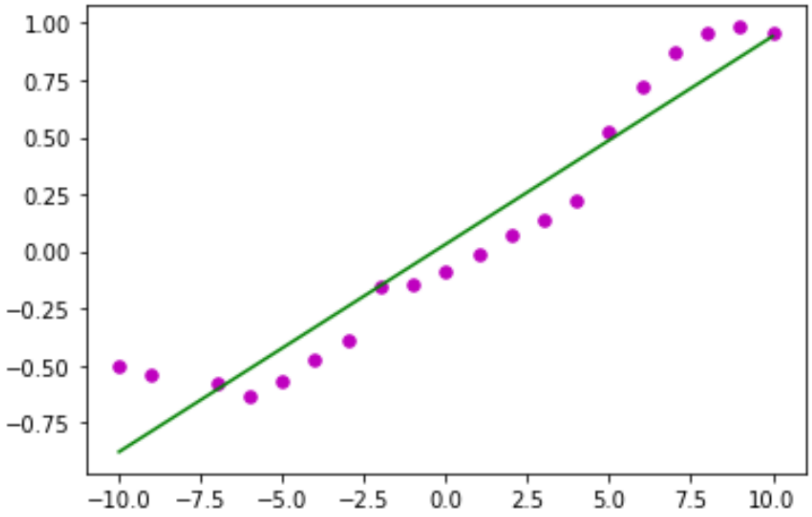


Figure 19: Sample Airfoil 3

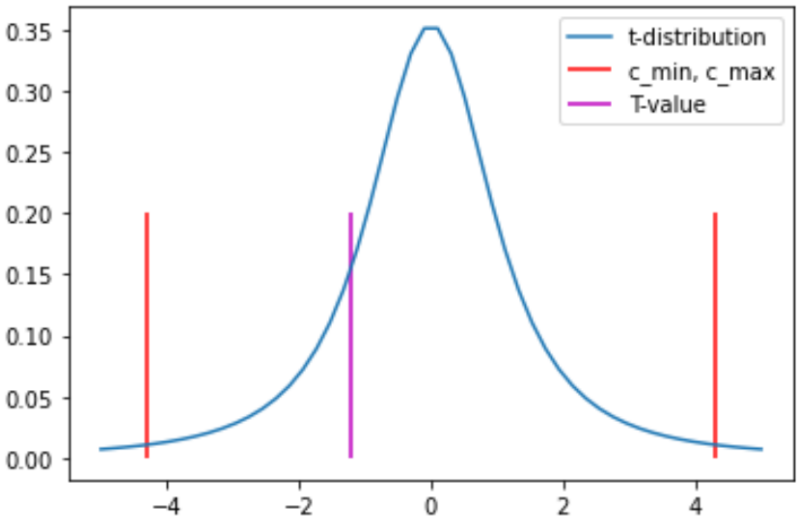


Figure 20: t-distribution plot

From the final figure, we can see that for our chosen level of significance, the T-value lies well outside the critical region. Thus, we fail to reject our null hypothesis.
This can supports the thin airfoil theory, and we can say that it is highly likely that the theory is true.

8 Summary

8.1 Summary

So, we are at the end of our project report and it is time to finally summarize what have we done so far. But before we do that, we will take a small detour to appreciate what we have truly learnt through this project technical things aside.
We started off the project as complete strangers, as just a bunch of random people who had little to no interaction prior to the project. Throughout the course of the project, we slowly got closer. Our team got stronger with each team meet we had, or with each group discussion we had on whatsapp. All of us can honestly say that we all truly enjoyed working with each other, and that this project has been a really rewarding experience for us in this regard.
The reason we chose this particular topic was because we wanted to actually see stuff we read about in books, and in this case learn about as a part of our course AE152, happen in real life. We wanted to match this bookish knowledge to practical observations. In that sense, again this project did not disappoint. It was exciting to see every plot match what we expected, to confirm every theory we've learnt as a part of aerodynamics.
Again, this does not mean we didn't face any hurdles on our journey. Data collection, in our case, posed the hardest challenge. Understanding Joint Distributions followed in close second. But in spite of all the hurdles we faced, each and every member of our team never gave up and continued to put in all the effort they could. We all used the team as a safety net to help each other with any difficulties which we faced.

Now, actually moving on the technical details of what we learnt throughout the project,

- We observed that the C_L tends to center around 0.8 with a rather large tail extending towards 0. On the other hand, C_D was much more "evenly" distributed with the majority of the values lying near 0.05.

- From the various scatter plots, we can clearly observe a series of "parallel" curves matching our expectations from the theory portion. C_L vs α plots showcase the somewhat linear relationship between them. The scatter plot of C_D vs C_L showcases the expected constant relationship between them.https://www.overleaf.com/project/60d88509bca7c9692a2a23b1

- We computed the 95% confidence interval of our random variables α, C_L and C_D using a random sample collected from the true population.

- We performed linear regression on α and C_L to verify the empirical linear relationship between them. We repeated a similar process for C_L and C_D to verify the expected relationship between them.

- We made a hypothesis based on the *thin-airfoil theory* and used the tools learnt in classes to state a null and alternate hypothesis, and *confirmed* our null hypothesis by calculating the critical region for a chosen level of significance(5%).

8.2 A message to the professors and the TAs

This marks the end of a beautiful 3.5 month journey which was the course AE102. This project has served as a platform for not only improvement in an academic sense, but overall development as a person as well. This project taught us how to work in a team, how to distribute work equally and fairly while using the strengths of each of our team member to max, how to make adjustments for each other, how to improvise when something goes wrong. We would like to extend a sincere thanks to the professors for giving us an amazing opportunity to work on a project which helps in the overall development of skills which will be helpful long after this course.

We would also like to thank the TAs for solving our doubts and giving us their support throughout the duration of this course.

This is team Airbenders, signing off.

9 Links and References

9.1 For airfoils coordinates data

UIUC AIRFOIL COORDINATES DATABASE
AIRFOIL TOOLS
From coordinates of airfoils to get data of α, C_L and C_D
XFOIL SOFTWARE

9.2 Data files

True population

Airfoil-wise data as collected by each member-
Airfoil Data set 1 Airfoil Data set 2 Airfoil Data set 3 Airfoil Data set 4
Airfoil Data set 5 Airfoil Data set 6

9.3 Video link

Link to video