Hochschule Flensburg University of Applied Sciences

Wind Energy Technology Institute

Basic Blade Design

Exercise for Lecture 01
Introduction to wind Turbine Aerodynamics

Presented by:

Hassan Pour Saeid
Mozafary Mostafa
Patel Manthan
Patil Rahul Bhatubhai
Soni Karan

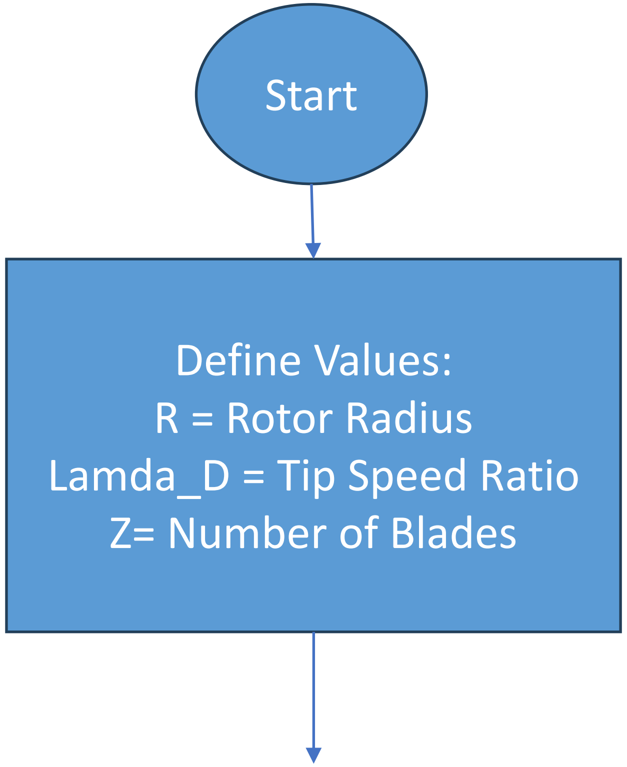
Overview

1. Tasks, Script and Output
2. Results
3. References

1. Tasks, Script and Output

a) Find the Design tip speed ration, rotor radius and number of blades.

Overview of Method used for Blade Design According Betz



Tip speed ratio (λD):

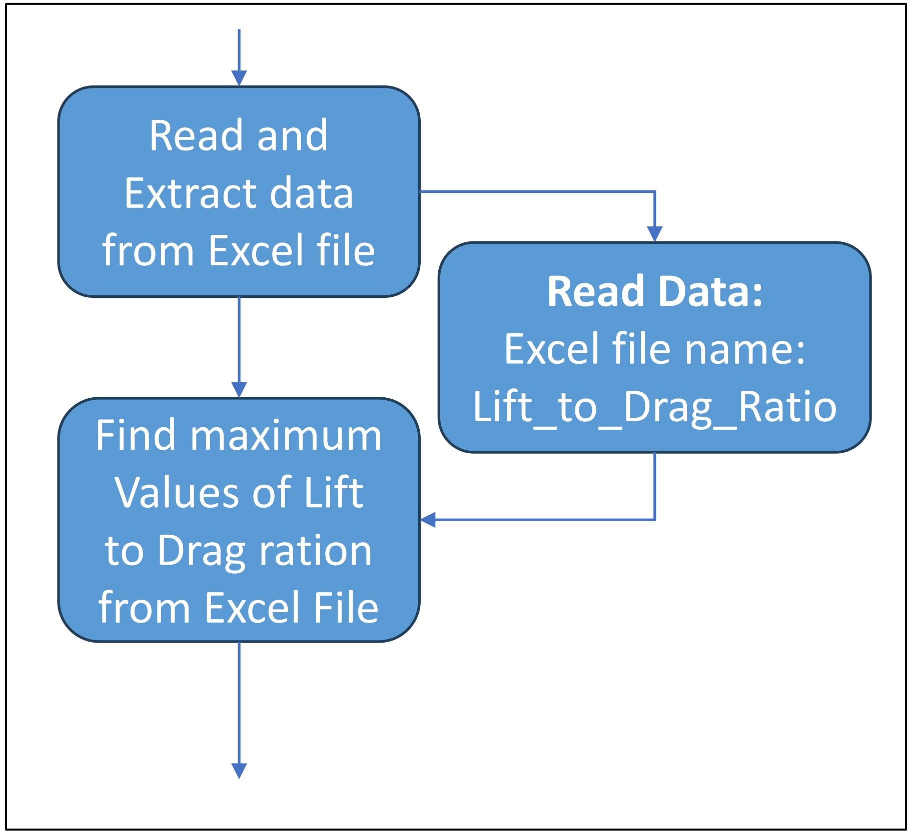
$$\lambda D = \frac{w\ x\ R}{v} = \frac{2\ x\ \pi\ x\ 12.1\ x\ 63}{60\ x\ 11.4} = 6.998 = 7$$

Script:

```
import numpy as np
import math
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# a) find R, lambda_D and z
# Define the Variables and Values
R        = 63           # [m]
Lamda_D  = 7            # [-]
Z        = 3            # [-]
```

1. Tasks, Script and Output

b) Find useful design values for angle of attack and lift coefficient. Use the airfoil data in file NACA64_A17.dat.



Lift to drag ratio (ε):

$$\varepsilon = \frac{C_L\,(\alpha_A)}{C_D\,(\alpha_A)}$$

C_L = Lift coefficient

C_D = Drag coefficient

α_A = Angle of attack

Script:

```
AirfoilFile =
pd.read_excel('C:/Users/s/Desktop/VS_Codes/Aerodynamic/Lift_to_Drag_Ratio.xlsx')
print (AirfoilFile)

# find maximum value of Lift to Drag_ratio
Maximum_Value_of_Lift_to_Drag_Ratio =
AirfoilFile['Lift to drag ratio'].max()
print (Maximum_Value_of_Lift_to_Drag_Ratio)
```

Output:

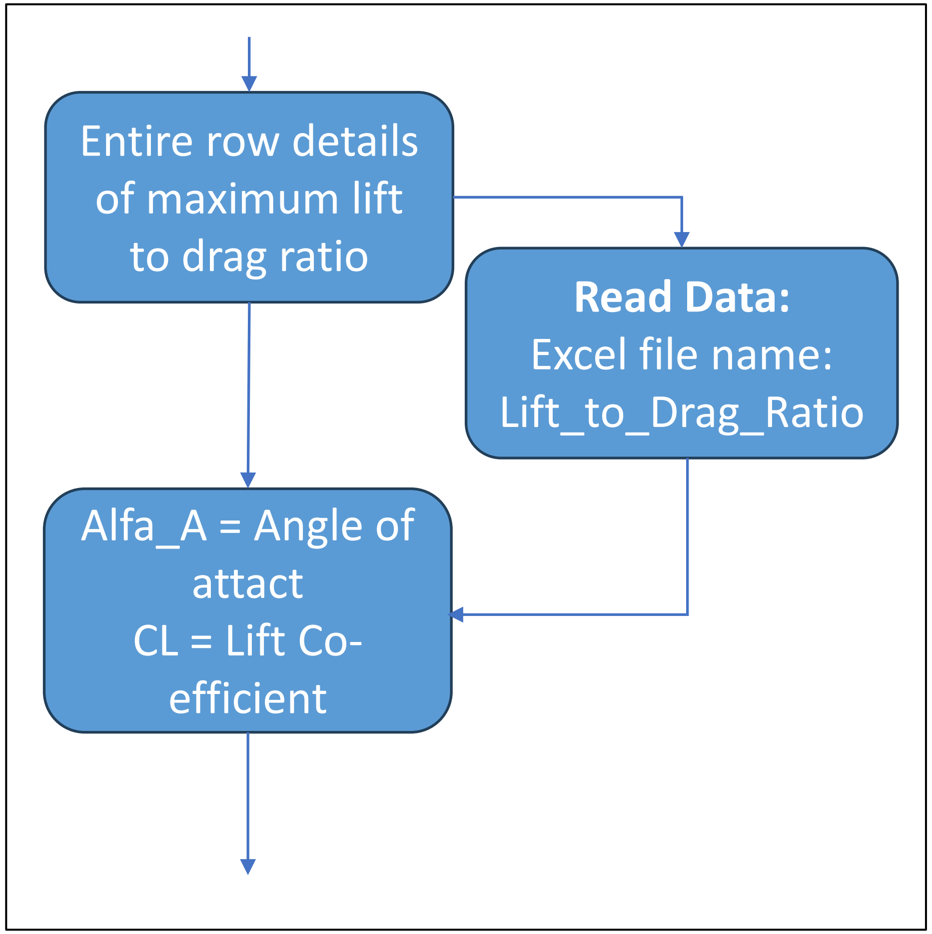
	Degree	Cl	Cd	Cm	Lift to drag ratio
0	-180.0	0.000	0.0198	0.0000	0.000000
1	-175.0	0.374	0.0341	0.1880	10.967742
2	-170.0	0.749	0.0955	0.3770	7.842932
3	-160.0	0.659	0.2807	0.2747	2.347702
4	-155.0	0.736	0.3919	0.3130	1.878030
..	...	...	...	...	...
122	155.0	-0.798	0.4116	-0.3349	-1.938776
123	160.0	-0.714	0.2931	-0.2942	-2.436029
124	170.0	-0.749	0.0971	-0.3771	-7.713697
125	175.0	-0.374	0.0334	-0.1879	-11.197605
126	180.0	0.000	0.0198	0.0000	0.000000

[127 rows x 5 columns]

174.3103448

1. Tasks, Script and Output

b) Find useful design values for angle of attack and lift coefficient. Use the airfoil data in file NACA64_A17.dat.



Script:

```
# Entire row details of maximum Lift to drag ratio
Row_Values = AirfoilFile[AirfoilFile['Lift to drag ratio'] == Maximum_Value_of_Lift_to_Drag_Ratio]
print(Row_Values)

# Take Alfa_A and CL directly from Row_Values
Alfa_A = Row_Values['Degree'].iloc[0]
CL = Row_Values['Cl'].iloc[0]
print(Alfa_A)
print(CL)
```

Output:

	Degree	Cl	Cd	Cm	Lift to drag ratio
61	5.0	1.011	0.0058	-0.124	174.310345

5.0

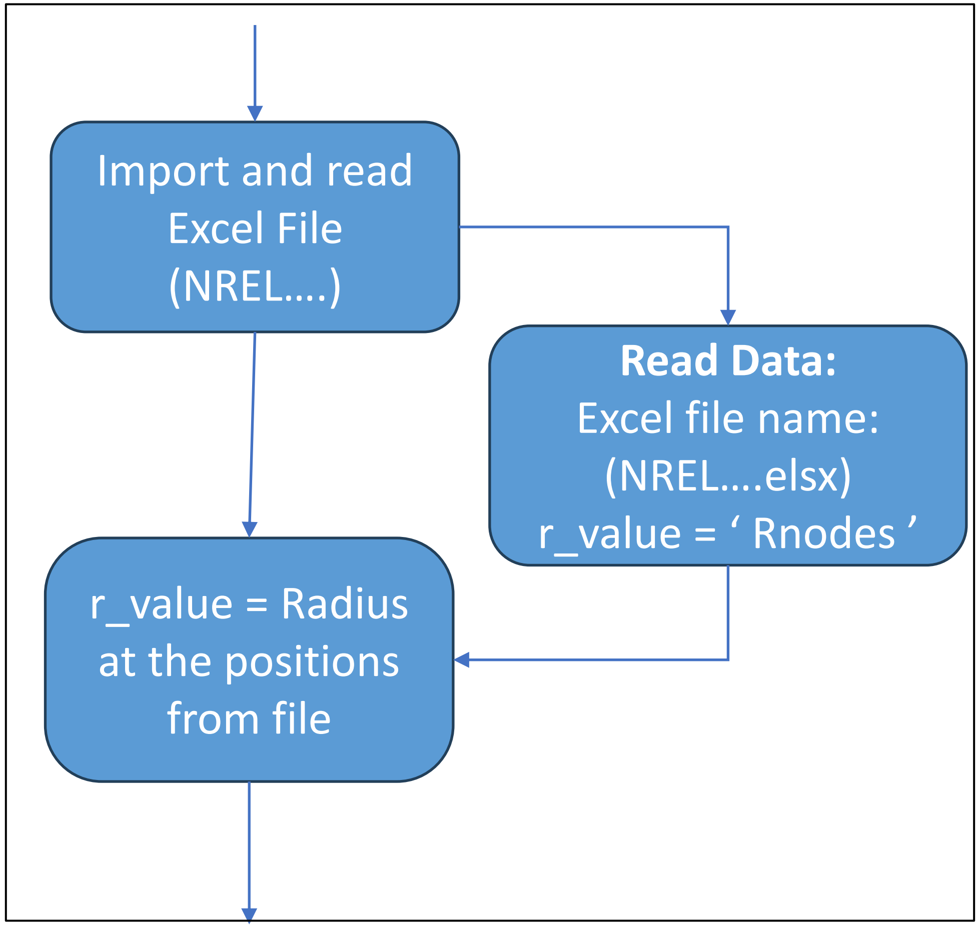
1.011

Hochschule Flensburg University of Applied Sciences

Wind Energy Technology Institute

1. Tasks, Script and Output

c) Calculate the distribution of twist angle β(r) and of chord length c(r) over the radius at the positions from the file NRELOffshrBsline5MW_AeroDyn_Equil_noTwr.dat.



Script:

```
56. NRELoffshrbsline5MW = pd.read_excel('C:/Users/s/Desktop/VS_Codes/Aerodynamic/NRELOffshrBsline5MW_AeroDyn_Equil_noTwr.xlsx’)
57. print (NRELoffshrbsline5MW)
58. # Extract RNodes values from Excel files
59. r_values = NRELoffshrbsline5MW['RNodes’]
60. print (r_values)
```

Output:

	RNodes	AeroTwst	DRNodes	Chord	NFoil	Unnamed: 4	PrnElm
0	2.8667	13.308	2.7333	3.542	1		NOPRINT
1	5.6000	13.308	2.7333	3.854	1		NOPRINT
2	8.3333	13.308	2.7333	4.167	2		NOPRINT
3	11.7500	13.308	4.1000	4.557	3		NOPRINT
4	15.8500	11.480	4.1000	4.652	4		NOPRINT
5	19.9500	10.162	4.1000	4.458	4		NOPRINT
6	24.0500	9.011	4.1000	4.249	5		NOPRINT
7	28.1500	7.795	4.1000	4.007	6		NOPRINT
8	32.2500	6.544	4.1000	3.748	6		NOPRINT
9	36.3500	5.361	4.1000	3.502	7		NOPRINT
10	40.4500	4.188	4.1000	3.256	7		NOPRINT
11	44.5500	3.125	4.1000	3.010	8		NOPRINT
12	48.6500	2.319	4.1000	2.764	8		NOPRINT
13	52.7500	1.526	4.1000	2.518	8		NOPRINT
14	56.1667	0.863	2.7333	2.313	8		NOPRINT
15	58.9000	0.370	2.7333	2.086	8		NOPRINT
16	61.6333	0.106	2.7333	1.419	8		NOPRINT

0	2.8667
1	5.6000
2	8.3333
3	11.7500
4	15.8500
5	19.9500
6	24.0500
7	28.1500
8	32.2500
9	36.3500
10	40.4500
11	44.5500
12	48.6500
13	52.7500
14	56.1667
15	58.9000
16	61.6333

1. Tasks, Script and Output

Distribution Twist angle:

$$\beta(r) = \tan^{-1}\left(\frac{2}{3}\,\frac{R}{r\,\lambda_D}\right) - \alpha_A$$

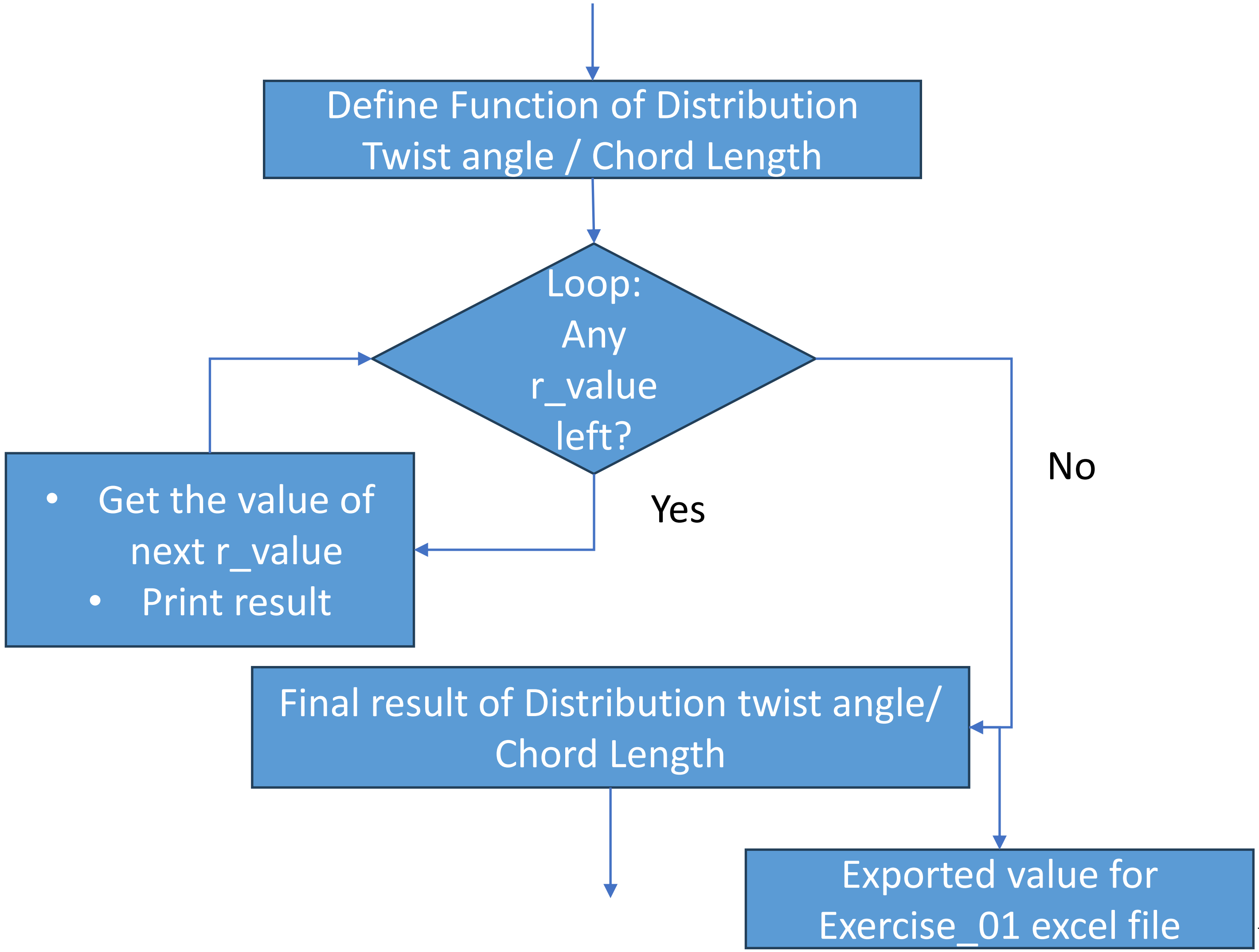
Distribution Chord Length:

$$c(r) = \frac{1}{2}\,\frac{8}{9}\,\frac{2\pi R}{C_L}\,\frac{1}{\lambda_D\sqrt{(\lambda_D\,\frac{r}{R})^2+4/9}}$$

Hochschule Flensburg University of Applied Sciences

Wind Energy Technology Institute

1. Tasks, Script and Output



Script :

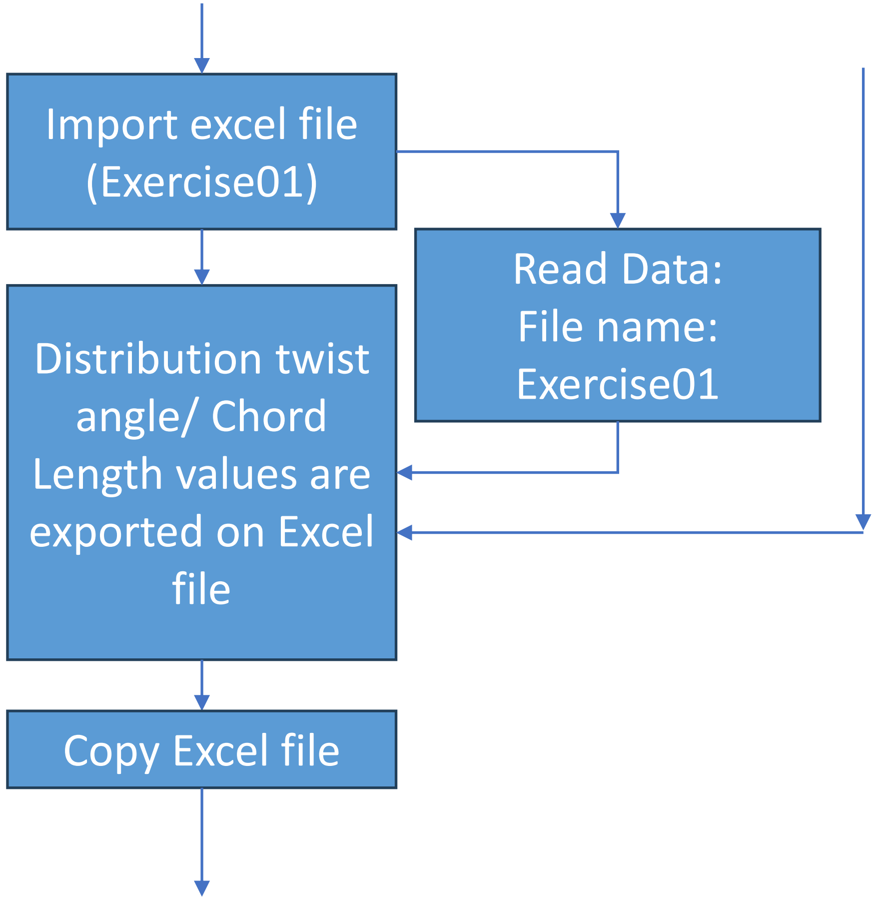
```
def Distrubution_Twist_Angle(R,r, Lamda_D, Alfa_A):
    return math.degrees(math.atan((2/3) * (R / (r * Lamda_D)))) - Alfa_A
# Define Loops for repait r values
result_Distrubution_Twist_angle = []
for r in r_values:
        result_D_T_A = Distrubution_Twist_Angle(R, r, Lamda_D, Alfa_A)
        print(f"For r = {r}: Result = {result_Distrubution_Twist_angle}")
        result_Distrubution_Twist_angle.append(result_D_T_
print(result_Distrubution_Twist_angle)
-----------------------------------------------------------------------------------------------------
# Define function Chord Length
def Chord_Length(R,r, Lamda_D, Z, CL):
    return (1/Z) * (8/9) * ((2 * math.pi * R)/CL) * (1/ (Lamda_D * ((Lamda_D * (r/R))**2 + (4/9) )** (1/2)))
result_Chord_Length = []
for r in r_values:
    r_C_Length = Chord_Length(R, r, Lamda_D, Z, CL)
    print(f"For r = {r}: Result = {result_Chord_Length}")
    result_Chord_Length.append(r_C_Length)
print(result_Chord_Length)
```

Output:

[59.462306457237204, 41.97493401088199, 30.753995929541922, 22.05059700708612, 15.734104734616874, 11.73875855842764, 9.00821226687864, 7.032191945895814, 5.539183728628229, 4.372837573824077, 3.4372357022852, 2.6704481043120047, 2.030779954414248, 1.4891663959275574, 1.097489986081471, 0.8165179873501573, 0.5602219024525228]

[22.430582895204765, 18.1735142241734, 14.525438337673771, 11.305436787606341, 8.80097374635135, 7.159683739189276, 6.017463537827824, 5.18219810960926, 4.546960757775722, 4.048539960826194, 3.647501845374648, 3.3180946102827256, 3.042839949409356, 2.809481131610511, 2.640566392625111, 2.5193176742611656, 2.4086645178811623]

1. Tasks, Script and Output



Script

```
# Read Exercise_01 excel file
Exercise_01 = pd.read_excel('C:/Users/s/Desktop/VS_Codes/Aerodynamic/Exercise01.xlsx')
print (Exercise_01)
# Chord length values are being exported for Exercise_01 excel file
for j in range(len(result_Chord_Length)):
Exercise_01.iloc[0 + j, 1] = result_Chord_Length[j]
# Distrubution Twist angle values are being exported for Exercise_01 excel file
for k in range(len(result_Distrubution_Twist_angle)):
Exercise_01.iloc[0 + k, 3] = result_Distrubution_Twist_angle[k]
print (Exercise_01)
# Updated Values back to the original Excel file (Exercise01_copy)
Exercise_01.to_excel('C:/Users/s/Desktop/VS_Codes/Aerodynamic/Exercise01_copy.xlsx')
```

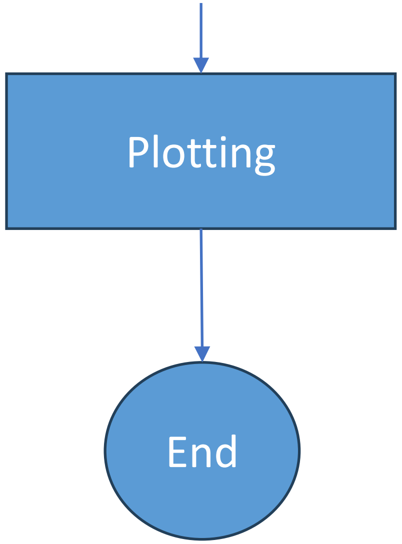
Output:

	r	Chord Betz	Chord NREL	Twist Betz	Twist NREL
0	2.8667	NaN	3.542	NaN	13.308
1	5.6000	NaN	3.854	NaN	13.308
2	8.3333	NaN	4.167	NaN	13.308
3	11.7500	NaN	4.557	NaN	13.308
4	15.8500	NaN	4.652	NaN	11.480
5	19.9500	NaN	4.458	NaN	10.162
6	24.0500	NaN	4.249	NaN	9.011
7	28.1500	NaN	4.007	NaN	7.795
8	32.2500	NaN	3.748	NaN	6.544
9	36.3500	NaN	3.502	NaN	5.361
10	40.4500	NaN	3.256	NaN	4.188
11	44.5500	NaN	3.010	NaN	3.125
12	48.6500	NaN	2.764	NaN	2.319
13	52.7500	NaN	2.518	NaN	1.526
14	56.1667	NaN	2.313	NaN	0.863
15	58.9000	NaN	2.086	NaN	0.370
16	61.6333	NaN	1.419	NaN	0.106

Update Excel file:

0	2.8667	22.430583	3.542	59.462306	13.308
1	5.6000	18.173514	3.854	41.974934	13.308
2	8.3333	14.525438	4.167	30.753996	13.308
3	11.7500	11.305437	4.557	22.050597	13.308
4	15.8500	8.800974	4.652	15.734105	11.480
5	19.9500	7.159684	4.458	11.738759	10.162
6	24.0500	6.017464	4.249	9.008212	9.011
7	28.1500	5.182198	4.007	7.032192	7.795
8	32.2500	4.546961	3.748	5.539184	6.544
9	36.3500	4.048540	3.502	4.372838	5.361
10	40.4500	3.647502	3.256	3.437236	4.188
11	44.5500	3.318095	3.010	2.670448	3.125
12	48.6500	3.042840	2.764	2.030780	2.319
13	52.7500	2.809481	2.518	1.489166	1.526
14	56.1667	2.640566	2.313	1.097490	0.863
15	58.9000	2.519318	2.086	0.816518	0.370
16	61.6333	2.408665	1.419	0.560222	0.106

1. Tasks, Script and Output

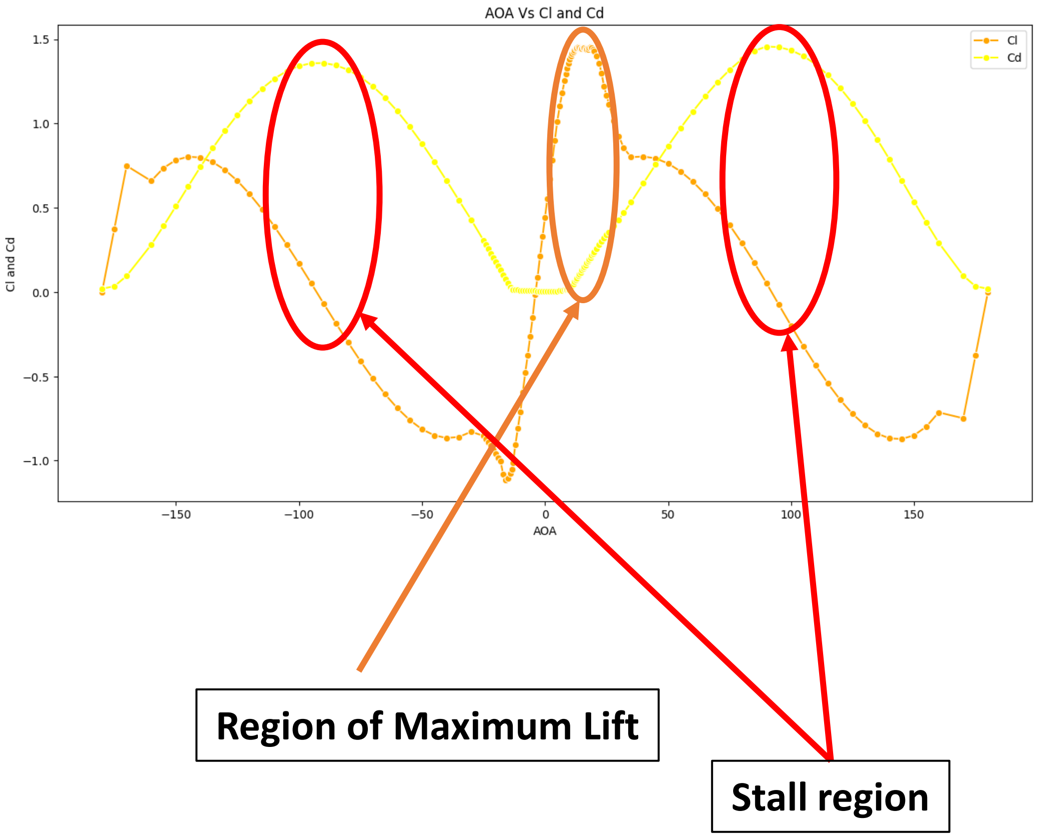


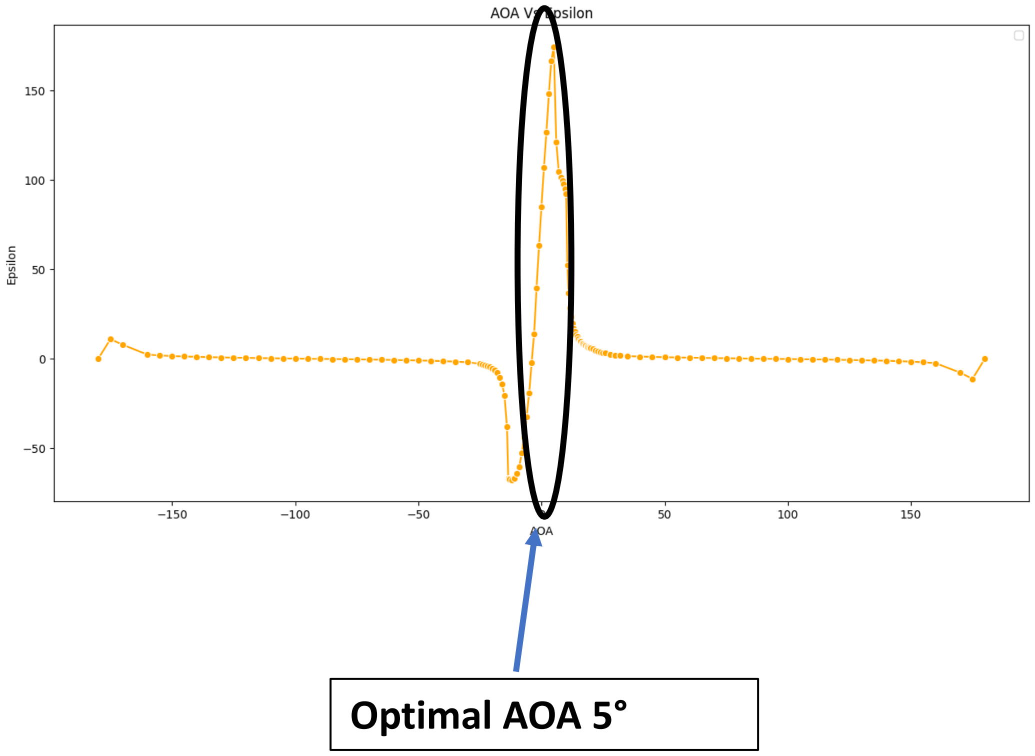
d) Compare your value to the ones from the NREL design.

Script

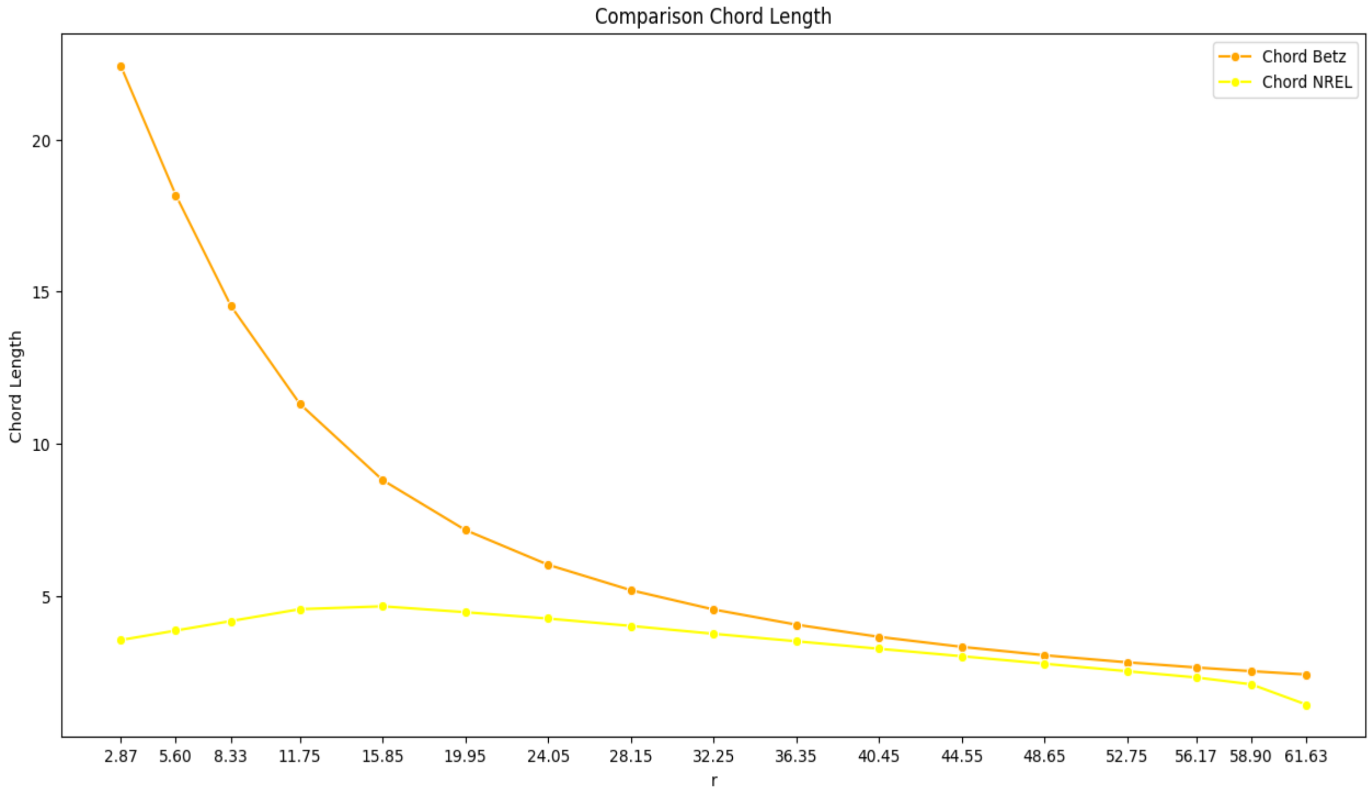
```
# Plotting Comparison of chord Length
plt.figure()
graph = sns.lineplot(x= 'r',y= 'Chord Betz', data = Exercise_01, color = 'Orange', marker = 'o', label = 'Chord Betz’)
graph = sns.lineplot(x= 'r',y= 'Chord NREL', data = Exercise_01, color = 'Yellow', marker = 'o', label = 'Chord NREL’)
plt.xticks([2.87,5.60,8.33,11.75,15.85,19.95,24.05,28.15,32.25,36.35,40.45,44.55,48.65,52.75,56.17,58.90,61.63])
plt.title('Comparison Chord Length’)
plt.xlabel('r’)
plt.ylabel('Chord Length’)
plt.legend()

# Plotting Comparison of Twist angle
plt.figure()
graph_2 = sns.lineplot(x= 'r',y= 'Twist Betz', data = Exercise_01, color = 'Orange', marker = 'o', label = 'Twist Betz’)
graph_2 = sns.lineplot(x= 'r',y= 'Twist NREL', data = Exercise_01, color = 'Yellow', marker = 'o', label = 'Twist NREL’)
plt.xticks([2.87,5.60,8.33,11.75,15.85,19.95,24.05,28.15,32.25,36.35,40.45,44.55,48.65,52.75,56.17,58.90,61.63])
plt.title('Comparison Twist Angle’)
plt.xlabel('r’)
plt.ylabel('Twist Angle’)
plt.legend()
# Show the plot
plt.show()
```

5.Results


AOA Vs Cl and Cd
Cl
Cd
Cl and Cd
AOA
Region of Maximum Lift
Stall region


Epsilon
AOA
Optimal AOA 5°

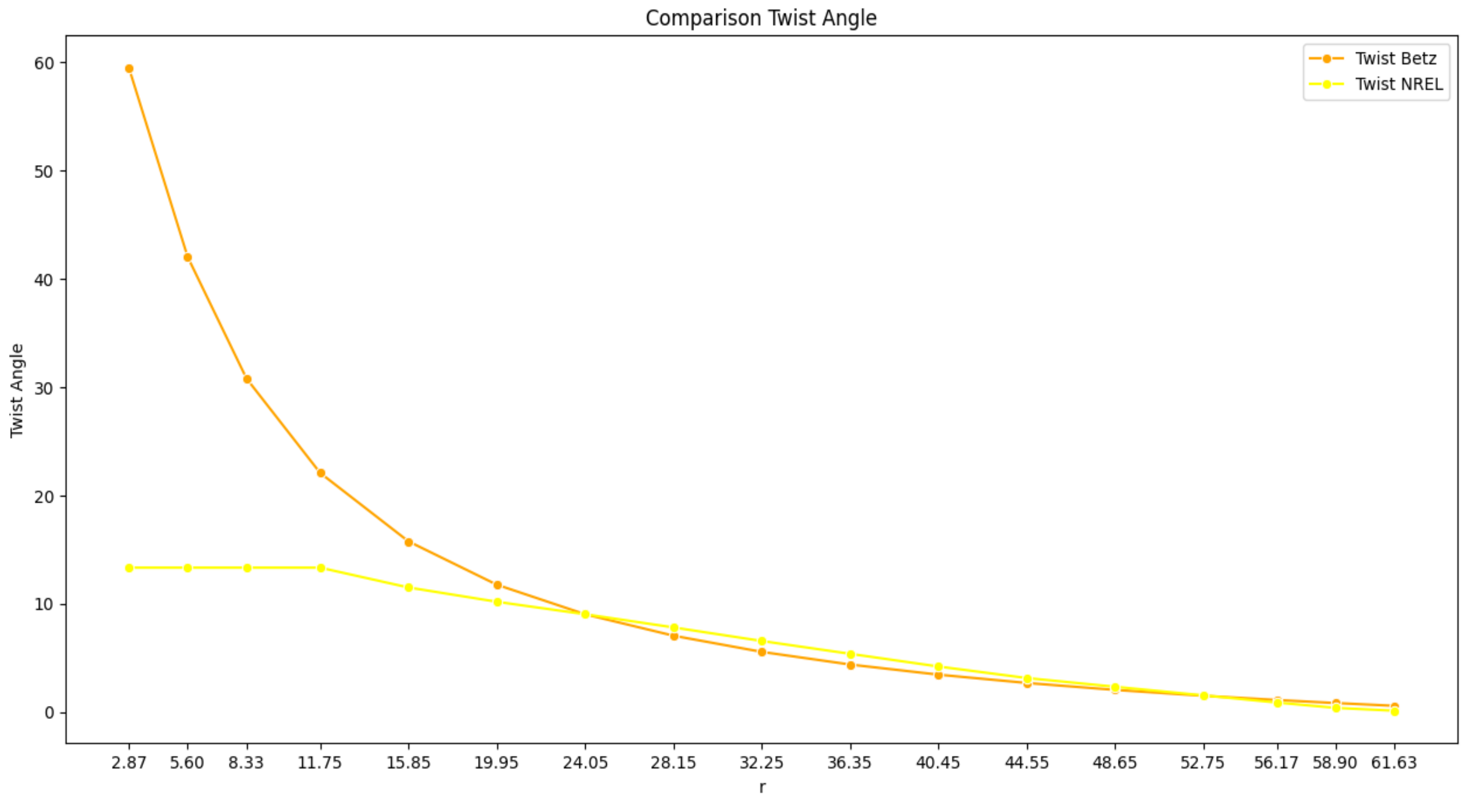
5.Results



- ✓ Increase in Aerodynamic Efficiency
- ✓ Enhance control and Stability
- ✓ Increased Structural integrity

- High cost of Manufacturing
- Increased Weight
- With increase in wind speed drag increases

5.Results



- ✓ Minimized Aerodynamic loses and improved lift
- ✓ Reduced stress concentration
- ✓ Improved response to turbulent wind conditions
- ✓ Enhanced yaw and pitch control

- Manufacturing complexity
- Increased stall effect

6. References

1. Prof. Dr.-Ing. David Schlipf. “Lecture #1 Introduction to Wind Turbine Aerodynamics” [Lecture notes]. 25.03.2024.
2. J. Jonkman, S. Butterfield, W. Musial, and G. Scott. Definition of a 5-MW Reference Wind Turbine for Offshore System Development. NREL, 2009.
3. Python Software Foundation. The Python Standard Library. From https://docs.python.org.
4. Prof. Ragunathan Rengasamy. “ Python for Data science, IIT Madras “ [Online Lecture]. https://nptel.ac.in.
5. Temesgen Batu & Hirpa G. Lemu(2020):Comparative Study of the Effect of Chord Length Computation Methods in Design of Wind Turbine Blade. Springer (((LNEE,volume 634)) https://link.springer.com/chapter/10.1007/978-981-15-2341-0_14
6. Mustafa Alaskari, Oday Abdullah & Mahir H. Majeed (2019): Analysis of Wind Turbine Using QBlade Software. Conference Series Materials Science and Engineering https://www.researchgate.net/publication/333661636