

**UTAA**

**University of Turkish Aeronautical Association**

**(AEE 497)**

**Python for Engineering Computations Midterm Report**

**Instructor: DOÇ. DR. MECİT YAMAN**

140121033 Emre Hüseyin Kamışlı

**02/05/2019**

# Table of the Content

[Table of the Content 1](#_Toc7623835)

[I. Part A) 3](#_Toc7623836)

[A. Normalizing the Coordinates 3](#_Toc7623837)

[II. Part B) 3](#_Toc7623838)

[A. Plot of the Airfoil 3](#_Toc7623839)

[B. Plot of the Mean Camber Line and the Chord Line 4](#_Toc7623840)

[C. Maximum Thickness and Position 5](#_Toc7623841)

[III. Part C) 6](#_Toc7623842)

[A. Panels and Panel Normals 6](#_Toc7623843)

[B. Cusped or Pointed 8](#_Toc7623844)

[IV. Finally 9](#_Toc7623845)

[A. References: 10](#_Toc7623846)

# Part A)

## Normalizing the Coordinates

The Coordinates should be fit in definite interval especially in x coordinates should start from 1 to 0 then from 0 to 1. After creating the desired x coordinates, we need to interpolate to get the corresponding y coordinates, note that interpolation is an approach, so we have some errors, the best point is we can create as many dots as we want. Now that we have x and y coordinates that accept the need, we can successfully normalize them.

In this project firstly we divide the x point and y point with fist value of the x point thus this process ıt was finalized x point start 1 and finish at again 1 then y values edited with this new values. Secondly, x coordinates should contain 0 on the middle so with Boolean type condition we create 0 value according to number of coordinates.

# Part B)

## Plot of the Airfoil

After the normalized coordinates with the matplotlib.pyplot library the figure plotted then the size and boundaries of the figure have been edited.

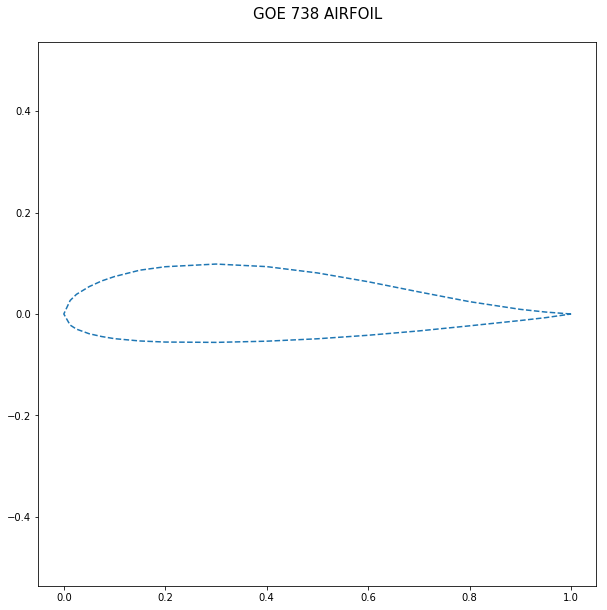


Figure Figure of Normalized Coordinates

## Plot of the Mean Camber Line and the Chord Line

The mean camber line (mcl) is join of the leading and trailing edges of an airfoil equidistant from the upper and lower surfaces. The mean camber line determines the characteristics of the airfoil. Also it is known as a camber line or a mean line.

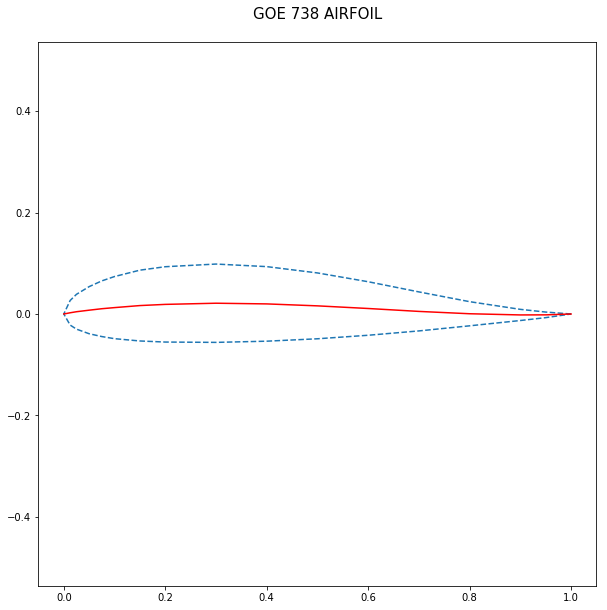


Figure Figure of Mean Camber Line

The Cord line is the straight line between leading edge and trailing edge. In the python code that is evaluated with max x value and min x value and with related y values then plot with these point the aid of matplotlib.pyplot library.

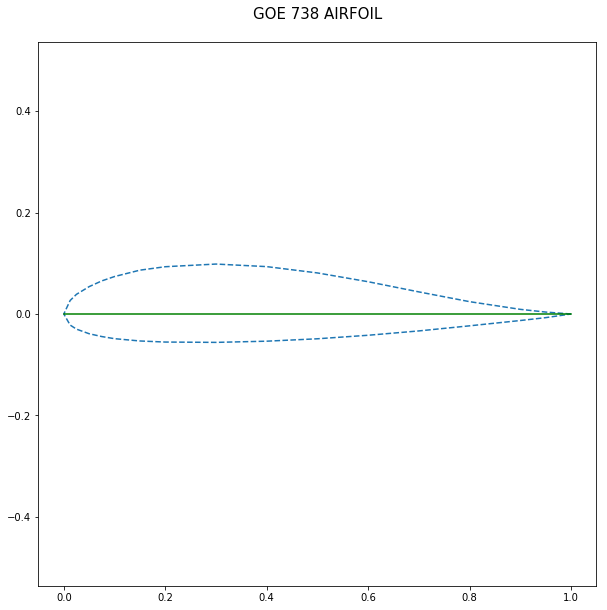


Figure Figure of Chord Line

## Maximum Thickness and Position

The maximum thickness (mt) is biggest length of the between upper and lower lines. In the python code that is found with a loop that is evaluated between upper and lower y values then it is continue one upper and one lower value by one by finally comparison the distances and stay on the biggest value.

Then location is equal the loop number with in the x value.

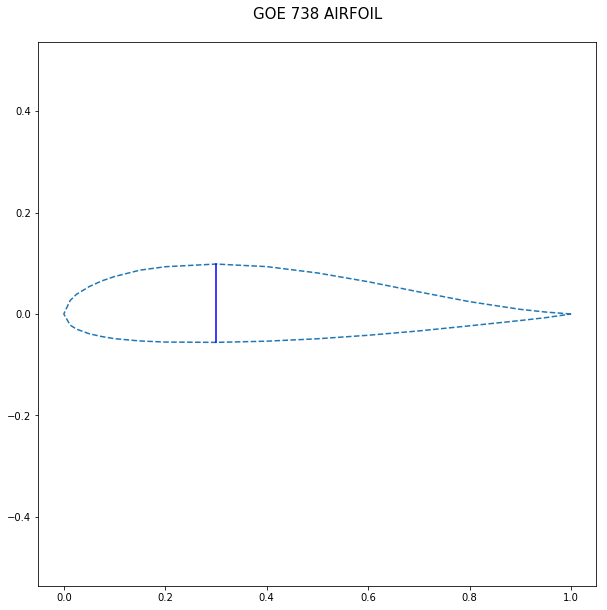


Figure Figure of Maximum Thickness

# Part C)

## Panels and Panel Normals

Panel methods refer to numerical methods for calculating the flow around any wing section. They are based on the replacement of the wing sections geometry by singularity panels. In this project 20 panels created then 20 normal drawn. In the python code firstly a circle had created around the airfoil shape then the circle is divided by the desired number of panels after this defined point on the circle projection of the x-coordinate on the surface. After this we reached the x point on the surface.

Secondly, in equation 2 a is slope already know the x and y value on the airfoil line and we can evaluate the b values with equation 1 and equation 2 that’s shown on the following equation

Finally, the desired projection x, a and b values are known so we can reach the new y values with aid of the equation 1.

On the normals first we had calculated the tangents with aid of the x and y desired values. Then we had found the normals with the magnitude of the normals in the end Normals are drawn with plt.quiver() command.

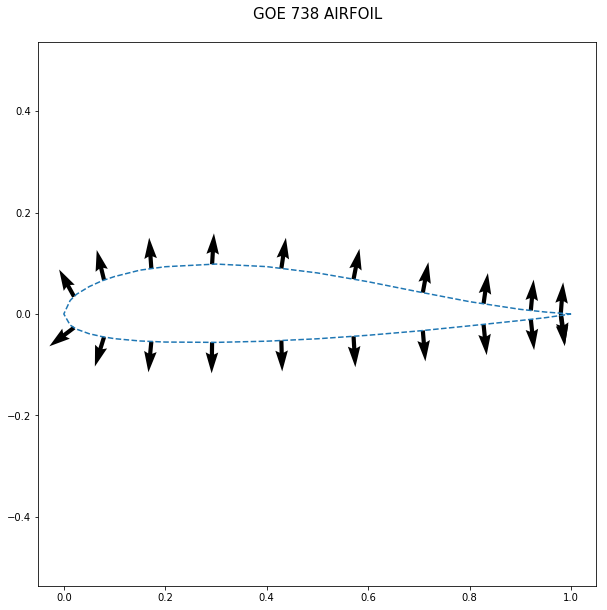


Figure Figure of Panel Normals

## Cusped or Pointed

The Kutta condition is a principle in steady-flow fluid dynamics, especially aerodynamics that is applicable to solid bodies with sharp corners, such as the trailing edges of airfoils.

The slopes are help us to defined the how is the airfoil type cusped or pointed if the training two line angle is higher than 15 degrees this airfoil shape is pointed other vise cusped.

In the following figure the finite angle (pointed) and cusped edge types are shown clearly, and in the python code this distinction defined with the if-else condition.

Figure Finite angle Edge

Figure Figure of Cusped Edge

# Finally

The combination of all shapes is shown in the following figure.

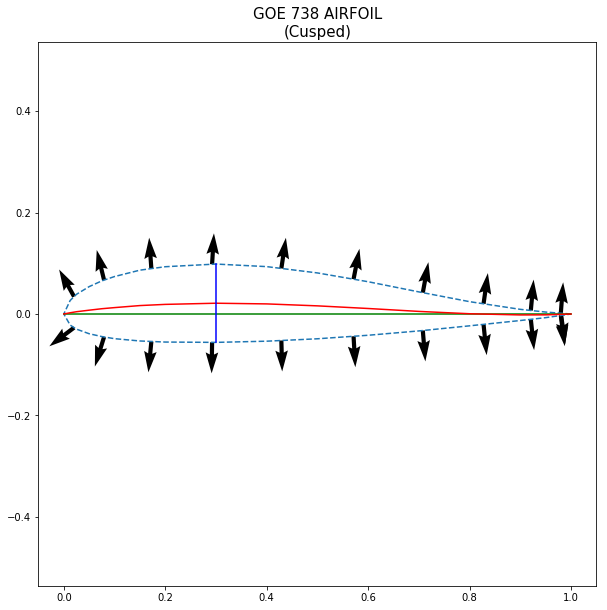


Figure Figure of Final Shape

## References:

* <https://nbviewer.jupyter.org/github/barbagroup/AeroPython/blob/master/lessons/10_Lesson10_sourcePanelMethod.ipynb>
* “4/5.” Fundamentals of Aerodynamics, by John David Anderson, McGraw-Hill Education, 2017.

# Appendix

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Sat Apr 13 10:57:56 2019
4. @author: Emre
5. """
6. import math
7. import numpy as np
8. import matplotlib.pyplot as plt
9. ascii
10. import os
11. import glob
12. **#Part A**
13. # =============================================================================
14. def normalize(x,y,name):
15. """
16. The Coordinates are fitting in definited interval
17. especially in x coordinates start from 1 to 0 then from 0 to 1.
19. arg: two arrays of x and y.
21. return: normalized x and y values
22. """
23. x=airfoil[:,0]/x[0]
24. y=airfoil[:,1]/x[0]
25. # make the zero of the x coordinates
26. if len(x)%2==0:
27. x[int((len(x))/2)-1]=0 #for dual total number of points
28. elif len(x)%2!=0:
29. x[int((len(x))/2)]=0 #for single total number of points
30. xnew=np.asarray([x,y])
31. xnew=np.transpose(xnew)
32. np.savetxt('C:/Users/Toshıba/Desktop/figures/datas/'+name[:-2]+'.txt',xnew)
33. return x, y
34. # =============================================================================
36. **#Part B**
37. # =============================================================================
38. def plot\_airfoil(x,y,name,status):
39. """
40. plotting airfoil surfaces and adjusting the boundarys
42. arg: two arrays of x and y and name of the airfoil
43. and status is the define the cusped or pointed
45. return : None
46. """
48. plt.figure(figsize= (8,6))
49. plt.ylim(ymax=.5)
50. plt.ylim(ymin=-.5)
51. # Name and Kutto conditoin type had wroten togetter on the up side.
52. title=(str(name)+str(status))
53. plt.title(title, fontsize=15)
54. plt.plot(x,y,'--')
55. plt.xlabel("X Coordinates",fontsize=15)
56. plt.ylabel("Y Coordinates",fontsize=15)
57. plt.axes().set\_aspect(1,'datalim')
58. # =============================================================================
60. # =============================================================================
61. def plot\_meancamberline(x,y):
62. """
63. plotting mean camber line that is middle line between upper an lower lines.
65. arg: two arrays of x and y.
67. return : None
68. """
69. mcly=[]
70. mclx=[]
71. # ran is the limit of the loop.
72. ran=(len(y)-1)/2
73. for i in range(int(ran)):
74. mcly.append((y[i]+y[-i-1])/2)
75. mclx.append((x[i]+x[-i-1])/2)
77. mcly.append(0)
78. mclx.append(0)
79. plt.plot(mclx,mcly,'r-')
80. plt.axes().set\_aspect(1,'datalim')
81. # =============================================================================
82. # =============================================================================
83. def plot\_chordline(x,y):
84. """
85. plotting the chord line that is line between fist and last x points.
87. arg: two arrays of x and y.
89. return : None
90. """
91. clx=[]
92. cly=[]
93. ran=(len(y)-1)/2
94. cly.append(y[0])
95. clx.append(x[0])
96. cly.append(y[int(ran)])
97. clx.append(x[int(ran)])
99. plt.plot(clx,cly,'g-')
100. # That is neccessary for the set the aspect ratios of the figure.
101. plt.axes().set\_aspect(1,'datalim')
102. # =============================================================================
103. # =============================================================================
104. def max\_thickness(x,y):
105. """
106. plotting the maximum thickness
107. the max distance between upper and lower surfaces.
109. arg: two arrays of x and y.
111. return : maximum thickness length.
112. """
113. mt=0
114. t=0
115. ran=(len(y)-1)/2
116. for i in range(int(ran)):
117. t=(y[i]-y[-i-1])
118. # in the following loop evaluate the maximum thickness for every loop
119. # if the previous is bigger than other stay same.
120. if t >= mt:
121. mt=t
122. location=i
124. mtx=[]
125. mty=[]
126. mty.append(y[location])
127. mtx.append(x[location])
128. mty.append(y[-location-1])
129. mtx.append(x[-location-1])
130. plt.text(0.55,0.35,"Maximum Thickness location: "+str(mt)+"c")
131. plt.plot(mtx,mty,'b-')
132. plt.axes().set\_aspect(1,'datalim')
133. return mt
134. # =============================================================================
135. **#Part C**
136. # =============================================================================
137. def plot\_panels(x,y,N):
138. """
139. plotting the panels normal with aid of the circle attibutes.
141. arg: two arrays of x,y and number of the panel number With N.
143. return : None.
144. """
145. N=N+2
146. R = (x.max() - x.min()) / 2 # radius of the circle
147. x\_center = (x.max() + x.min()) / 2 # x-coord of the center
148. # define x-coord of the circle points
149. x\_circle = x\_center + R \* np.cos(np.linspace(0.0, 2 \* math.pi, N +1))
150. x\_ends = np.copy(x\_circle) # projection of the x-coord on the surface
151. y\_ends = np.empty\_like(x\_ends) # initialization of the y-coord Numpy array
153. #x, y = np.append(x, x[0]), np.append(y, y[0]) # extend arrays using numpy.append
155. I = 0
156. for i in range(N):
157. while I < len(x) :
158. if (x[I] <= x\_ends[i] <= x[I + 1]) or (x[I + 1] <= x\_ends[i] <= x[I]):
159. break
160. else:
161. I += 1
162. a = (y[I + 1] - y[I]) / (x[I + 1] - x[I]) # a is the slope
163. b = y[I + 1] - a \* x[I + 1] # b is evaluating with aid of the a.
164. y\_ends[i] = a \* x\_ends[i] + b
165. y\_ends[N] = y\_ends[0]
167. tx=x\_ends[1:]-x\_ends[:-1]
168. ty=y\_ends[1:]-y\_ends[:-1]
169. nx=ty
170. ny=-tx
171. n\_ = (nx\*\*2+ny\*\*2)\*\*.5
172. nx=nx/n\_
173. ny=ny/n\_
175. #Create list from the array for using pop command.
176. x\_ends=list(x\_ends)
177. y\_ends=list(y\_ends)
178. nx=list(nx)
179. ny=list(ny)
180. #Remove the first and end point for does not create normal.
181. x\_ends.pop()
182. y\_ends.pop()
183. y\_ends.pop(0)
184. x\_ends.pop(0)
185. x\_ends.pop(int(N/2)-1)
186. y\_ends.pop(int(N/2)-1)
187. nx.pop(int(N/2)-1)
188. ny.pop(int(N/2)-1)
190. plt.quiver(x\_ends,y\_ends,nx,ny)
191. return
192. # =============================================================================
193. # =============================================================================
194. def is\_cusped(x,y):
195. """
196. Evaluating the is this airfoil Cusped or Pointed
198. arg: two arrays of x and y.
200. return : type of the trailing edge with string type.
201. """
202. # Exceed of the limit it is becoming pointed edge.
203. limit = np.deg2rad(15)
204. upper\_slope = (y[0]-y[1])/(x[0]-x[1])
205. lower\_slope = (y[-1]-y[-2])/(x[-1]-x[-2])
207. # theta is the angle between upper and lower trailing edge surfaces.
208. theta =np.arctan(upper\_slope)-np.arctan(lower\_slope)
210. if abs(theta) <= limit:
211. status=('(Cusped)')
212. else:
213. status=('(Pointed)')
214. return status
215. # =============================================================================

**#Part Main Part**

1. # =============================================================================
2. airfoil='naca1408.dat'
3. name=open(airfoil,"r")
4. name=name.readline()
5. airfoil = np.loadtxt(airfoil, skiprows=1)
6. x,y=airfoil[:,0],airfoil[:,1]
7. if 1==1:
8. N=20
9. # get the normalized coordinates as x and y.
10. x,y=normalize(x,y,name)
11. # get cusped or pointed with string type.
12. status=is\_cusped(x,y)
13. # plot the airfoil using x, y coordinates
14. plot\_airfoil(x,y,name,status)
15. # plot the chord line
16. plot\_chordline(x,y)
17. # plot the mean camberline
18. plot\_meancamberline(x,y)
19. # calculate and show max thickness of the airfoil
20. mt=max\_thickness(x,y)
21. plt.savefig('C:/Users/Toshıba/Desktop/figures/without panel/'+ name[:-2] +'.png')
22. plot\_panels(x,y,N)
23. if 2==2:
24. N=20
25. # get the normalized coordinates as x and y.
26. x,y=normalize(x,y,name)
27. # get cusped or pointed with string type.
28. status=is\_cusped(x,y)
29. # plot the airfoil using x, y coordinates
30. plot\_airfoil(x,y,name,status)
31. # plot the chord line
32. plot\_chordline(x,y)
33. # plot the mean camberline
34. plot\_meancamberline(x,y)
35. # calculate and show max thickness of the airfoil
36. mt=max\_thickness(x,y)
37. plot\_panels(x,y,N)
38. plt.savefig('C:/Users/Toshıba/Desktop/figures/with panel/'+ name[:-2] +'.png')
40. # =============================================================================