

```

1 # coding: utf-8
2 __author__ = 'Geoffrey Nyaga'
3
4
5 import numpy as np
6 import math
7 import matplotlib.pyplot as plt
8
9 def llt(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0):
10
11     b = math.sqrt(AR*S) # wing span (m)
12     MAC = S/b # Mean Aerodynamic Chord (m)
13     Croot = (1.5*(1+taper)*MAC)/(1+taper+taper**2) # root chord (m)
14     # theta = np.arange(math.pi/(2*N), math.pi/2, math.pi/(2*(N)))
15     theta = np.linspace((math.pi/(2*N)), (math.pi/2),N,endpoint=True)
16     # alpha =np.arange(i_w+alpha_twist,i_w ,-alpha_twist/(N))
17     alpha = np.linspace(i_w+alpha_twist,i_w ,N)
18     z = (b/2)*np.cos(theta)
19     c = Croot * (1 - (1-taper)* np.cos(theta)) # Mean Aerodynamics
20     mu = c * a_2d / (4 * b)
21
22     LHS = (mu * (np.array(alpha)-alpha_0)/57.3)#.reshape((N-1),1)# Left Hand Side
23
24     RHS = []
25     for i in range(1,2*N+1,2):
26         RHS_iter = (np.sin(i*theta)*(1+(mu*i)/(np.sin(list(theta))))).reshape(1,N)
27         # print(RHS_iter,"RHS_iter shape")
28         RHS.append(RHS_iter)
29
30     test = (np.asarray(RHS))
31     x = np.transpose(test)
32     inv_RHS = np.linalg.inv(x)
33     ans = np.matmul(inv_RHS,LHS)
34
35     mynum = np.divide((4*b),c)
36     test = ((np.sin((1)*theta))*ans[0]*mynum)
37     test1 = ((np.sin((3)*theta))*ans[1]*mynum)
38     test2 = ((np.sin((5)*theta))*ans[2]*mynum)
39     test3 = ((np.sin((7)*theta))*ans[3]*mynum)
40     test4 = ((np.sin((9)*theta))*ans[4]*mynum)
41     test5 = ((np.sin((11)*theta))*ans[5]*mynum)
42     test6 = ((np.sin((13)*theta))*ans[6]*mynum)
43     test7 = ((np.sin((15)*theta))*ans[7]*mynum)
44     test8 = ((np.sin((17)*theta))*ans[8]*mynum)
45
46     CL = test+test1+test2+test3+test4+test5+test6+test7+test8
47     CL1 = np.append(0,CL)
48     y_s=[b/2 , z[0], z[1], z[2] ,z[3], z[4] ,z[5], z[6], z[7] ,z[8]]
49     # print(y_s,"ys")
50
51     if __name__ == "__main__":
52         plt.plot(y_s,CL1, marker='o')
53         plt.title('Lifting Line Theory\n Elliptical Lift distribution')
54         plt.xlabel('Semi-span location (m)')
55         plt.ylabel('Lift coefficient')
56         plt.grid()
57         if __name__ == "__main__":
58             plt.show()
59
60
61     CL_wing = math.pi * AR * ans[0] # USE THIS CL WITH CRUISE SPEED TO CALCULATE THE ACCURATE LIFT
62     !!!!!!!!!!!
63     myfinalmat = np.array(y_s)
64     myfinalmat2 = CL1
65
66     return myfinalmat,myfinalmat2, CL_wing #CL_wing,y_s,CL1
67
68     if __name__ == "__main__":
69         print (CL_wing,"CL_wing")
70
71 def llt_with_plots(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0):
72
73     b = math.sqrt(AR*S) # wing span (m)
74     MAC = S/b # Mean Aerodynamic Chord (m)
75     Croot = (1.5*(1+taper)*MAC)/(1+taper+taper**2) # root chord (m)
76     # theta = np.arange(math.pi/(2*N), math.pi/2, math.pi/(2*(N)))
77     theta = np.linspace((math.pi/(2*N)), (math.pi/2),N,endpoint=True)

```

```

77 # alpha = np.arange(i_w+alpha_twist,i_w,-alpha_twist/(N))
78 alpha = np.linspace(i_w+alpha_twist,i_w,N)
79 z = (b/2)*np.cos(theta)
80 c = Croot * (1 - (1-taper)* np.cos(theta)) # Mean Aerodynamics
81 mu = c * a_2d / (4 * b)
82
83 LHS = (mu * (np.array(alpha)-alpha_0)/57.3)#.reshape((N-1),1)# Left Hand Side
84
85 RHS = []
86 for i in range(1,2*N+1,2):
87     RHS_iter = (np.sin(i*theta)*(1+(mu*i)/(np.sin(list(theta)))))#.reshape(1,N)
88     # print(RHS_iter,"RHS_iter shape")
89     RHS.append(RHS_iter)
90
91 test = (np.asarray(RHS))
92 x = np.transpose(test)
93 inv_RHS = np.linalg.inv(x)
94 ans = np.matmul(inv_RHS,LHS)
95
96 mynum = np.divide((4*b),c)
97 test = ((np.sin((1)*theta))*ans[0]*mynum)
98 test1 = ((np.sin((3)*theta))*ans[1]*mynum)
99 test2 = ((np.sin((5)*theta))*ans[2]*mynum)
100 test3 = ((np.sin((7)*theta))*ans[3]*mynum)
101 test4 = ((np.sin((9)*theta))*ans[4]*mynum)
102 test5 = ((np.sin((11)*theta))*ans[5]*mynum)
103 test6 = ((np.sin((13)*theta))*ans[6]*mynum)
104 test7 = ((np.sin((15)*theta))*ans[7]*mynum)
105 test8 = ((np.sin((17)*theta))*ans[8]*mynum)
106
107 CL = test+test1+test2+test3+test4+test5+test6+test7+test8
108 CL1 = np.append(0,CL)
109 y_s=[b/2, z[0], z[1], z[2], z[3], z[4], z[5], z[6], z[7], z[8]]
110
111 # num1 = str(round(alpha_twist,2))
112 num1 = "{0:.2f}".format(alpha_twist)
113 num2 = "{0:.2f}".format(i_w)
114
115 plt.plot(y_s,CL1, marker='o',label = ( "alpha_twist:",num1,"wing_incidence:",num2 ) )
116 plt.title('Lifting Line Theory\n Elliptical Lift distribution')
117 plt.xlabel('Semi-span location (m)')
118 plt.ylabel('Lift coefficient')
119 plt.legend()
120 plt.grid()
121 # plt.show()
122
123 CL_wing = math.pi * AR * ans[0] # USE THIS CL WITH CRUISE SPEED TO CALCULATE THE ACCURATE LIFT
124 #!!!!!!
125 return CL_wing
126
127 if __name__ == "__main__":
128     print (CL_wing,"CL_wing")
129
130 def ll_t_subplots(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0):
131
132     b = math.sqrt(AR*S) # wing span (m)
133     MAC = S/b # Mean Aerodynamic Chord (m)
134     Croot = (1.5*(1+taper)*MAC)/(1+taper+taper**2) # root chord (m)
135     # theta = np.arange(math.pi/(2*N), math.pi/2, math.pi/(2*(N)))
136     theta = np.linspace((math.pi/(2*N)), (math.pi/2),N,endpoint=True)
137     # alpha = np.arange(i_w+alpha_twist,i_w,-alpha_twist/(N))
138     alpha = np.linspace(i_w+alpha_twist,i_w,N)
139     z = (b/2)*np.cos(theta)
140     c = Croot * (1 - (1-taper)* np.cos(theta)) # Mean Aerodynamics
141     mu = c * a_2d / (4 * b)
142
143     LHS = (mu * (np.array(alpha)-alpha_0)/57.3)#.reshape((N-1),1)# Left Hand Side
144
145     RHS = []
146     for i in range(1,2*N+1,2):
147         RHS_iter = (np.sin(i*theta)*(1+(mu*i)/(np.sin(list(theta)))))#.reshape(1,N)
148         # print(RHS_iter,"RHS_iter shape")
149         RHS.append(RHS_iter)
150
151     test = (np.asarray(RHS))
152     x = np.transpose(test)

```

```

153     inv_RHS = np.linalg.inv(x)
154     ans = np.matmul(inv_RHS,LHS)
155
156     mynum = np.divide((4*b),c)
157     test = ((np.sin((1)*theta))*ans[0]*mynum)
158     test1 = ((np.sin((3)*theta))*ans[1]*mynum)
159     test2 = ((np.sin((5)*theta))*ans[2]*mynum)
160     test3 = ((np.sin((7)*theta))*ans[3]*mynum)
161     test4 = ((np.sin((9)*theta))*ans[4]*mynum)
162     test5= ((np.sin((11)*theta))*ans[5]*mynum)
163     test6 = ((np.sin((13)*theta))*ans[6]*mynum)
164     test7 = ((np.sin((15)*theta))*ans[7]*mynum)
165     test8 = ((np.sin((17)*theta))*ans[8]*mynum)
166
167     CL = test+test1+test2+test3+test4+test5+test6+test7+test8
168     CL1 = np.append (0,CL)
169     y_s=[b/2 , z[0], z[1], z[2] ,z[3], z[4] ,z[5], z[6], z[7] ,z[8]]
170
171     # num1 =      str(round(alpha_twist,2))
172     num1 = "{0:.2f}".format(alpha_twist)
173     num2 = "{0:.2f}".format(i_w)
174
175     CL_wing = math.pi * AR * ans[0] # USE THIS CL WITH CRUISE SPEED TO CALCULATE THE ACCURATE LIFT
176     !!!!!!!!
177     return CL_wing
178
179     if __name__ == "__main__":
180         print (CL_wing,"CL_wing")

```