```
# coding: utf-8
__author__ = 'Geoffrey Nyaga'

import sys
sys.path.append('../')
from API.db_API import write_to_db, read_from_db

"""
  KENYA ONE PROJECT
          Matlab code to solve for CL of the wing and elliptical#
        lift distribution with flaps.
          -------------------------------------------------------
                  done by Geoffrey Nyaga Kinyua
          -------------------------------------------------------

"""

import numpy as np
import math
import matplotlib.pylab as plt

import lifting_line_theory as llt

N = llt.N  # (number of segments - 1)
S = llt.S # m^2
AR = llt.AR # Aspect ratio
taper = llt.taper # Taper ratio
alpha_twist = llt.alpha_twist # Twist angle (deg)
a_2d = llt.a_2d # lift curve slope (1/rad)
alpha_0 = llt.alpha_0 # zero-lift angle of attack (deg)
i_w = llt.i_w # wing setting angle (deg)

a_0 = -4.2 # flap up zero-lift angle of attack (deg)
fuselage_angle = 10
cfc=0.20
df=14

bf_b=0.6 #flap-to-wing span ratio

def llt_full(fuselage_angle,df):

  doflap=-1.15*cfc*df
  i_wing = fuselage_angle + i_w ; # takeoff_wing_setting_angle (deg)(fuselage angle+wing incidence)
  b = np.sqrt(AR*S)# wing span (m)
  a_0_fd = doflap+a_0 # flap down zero-lift angle of attack (deg)
  MAC = S/b # Mean Aerodynamic Chord (m)
  Croot = (1.5*(1+taper)*MAC)/(1+taper+taper**2) # root chord (m)
  theta = np.linspace((math.pi/(2*N)), (math.pi/2),N,endpoint=True)
  alpha = np.linspace(i_wing+alpha_twist,i_wing ,N)

  # segment’s angle of attack
  for i in range (1,N):
      if (i/N)>(1-bf_b):
          alpha_0=a_0_fd #flap down zero lift AOA

      else:
          alpha_0=a_0 #flap up zero lift AOA

  z = (b/2)*np.cos(theta)
  c = Croot * (1 - (1-taper)* np.cos(theta)) # Mean Aerodynamics
  mu = c * a_2d / (4 * b)
  LHS = (mu * (np.array(alpha)-alpha_0)/57.3)#.reshape((N-1),1)# Left Hand Side

  # Solving N equations to find coefficients A(i):
  RHS = []
  for i in range(1,2*N+1,2):
      RHS_iter = (np.sin(i*theta)*(1+(mu*i)/(np.sin(list(theta)))))#.reshape(1,N)
      RHS.append(RHS_iter)

  test = (np.asarray(RHS))
  x = np.transpose(test)
  inv_RHS = np.linalg.inv(x)
  ans = np.matmul(inv_RHS,LHS)

  CL_TO = math.pi * AR * ans[0]

  return CL_TO
```

```python

  # USE THIS CL WITH take-off speed TO CALCULATE THE ACCURATE LIFT!!!!!!!!!!

v_takeoff = 1.2*read_from_db('stallSpeed')*1.688

initial_CL_TO = (2*read_from_db('finalMTOW'))/(read_from_db('rhoSL')*v_takeoff**2 * S * 10.764)
print(initial_CL_TO,"initial_CL_TO")


def llt_final():

  fuselage_angle = np.arange(5,13,.1)
  df = np.arange(0,16,.1)

  x = []
  for i in fuselage_angle:
    for j in df:
      lst = (i,j)
      x.append (lst)

  finalyy = []
  mycombination = []
  outputcombination = []
  for fuselage_angle,df in x:
    final = llt_full(fuselage_angle,df)


    if abs( (final/initial_CL_TO) - 1 ) <= 0.0001:
      if __name__ == '__main__':
        print ("Calculated CL_TO",final,"possible combination",fuselage_angle,"<fuselage angle",df," match")
      num1 = "{0:.2f}".format(fuselage_angle)
      num2 = "{0:.2f}".format(df)
      mycombination.append((num1,num2))
      outputcombination.append((fuselage_angle,df))
      myy = (final)
      finalyy.append(myy)
      yy = (np.asarray(finalyy))
    else:
      pass
  print(mycombination,"select one ")
  return outputcombination


#select one combination
combination = llt_final()
selectone = 0 #########AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
mycombination = (combination[selectone])

print (mycombination,"final fuselage_angle and df")


llt_full(mycombination[0],mycombination[1])


write_to_db('CL_TO',initial_CL_TO)
```