```
######################################### UNCHANGED TASK 1 CODE #################################################
from scipy.optimize import fsolve
import numpy as np
from sympy import (symbols, sin, cos, solve, pi)
import math
a_0 = symbols('a_0')
T_0 = symbols('T_0')
i_h = symbols('i_h')
#Variables

W = 22000 #lb
A_in = 15 # sqft

cw_w=11.86
cw_h=6.71
cw_v=7.32

X_ac_wf = 28.0352 #ft
X_ac_h = 42.2825 #ft
X_ac_w = 28.628 #ft
X_cg = 27 #ft

S_H = 100 #ft^2
S_W = 400 #ft^2
A_in = 15 #ft^2

n_H = 0.5
rho=7.38*(10**-4) # slug/ft^3
q_inf = (1/2) * rho * (503.36 ** 2) #whatever units pressure is
#print(q_inf)
q_H = n_H * q_inf
vv=503.36 #ft/s

CL_a_wf = 3.1373
CL_a_H = 3.277
CL_a_W = 3.1373 #5.7875

de_da = 0.75

# Function representing the system of equations

def equations(variables):
    a_0, i_h, T_0 = variables

    e = de_da * a_0
    a_H = a_0 + i_h - e
    L_wf = CL_a_W * a_0 * q_inf * S_W #CL_a_wf is the same as CL_a_w
    L_h = CL_a_H * a_H * q_H * S_H
    C_D0 = 0.0145 + 0.1*((CL_a_W*a_0)**2)
    D_0 = q_inf * S_W * C_D0
    F_N0 = 2 * q_inf * A_in * (np.cos(a_0)**2) * np.sin(a_0)

    # Trim equations
    #X = T_0 - W*np.sin(a_0) + L_wf * np.sin(a_0) - D_0 *np.cos(a_0) + L_h*np.sin(a_0 - e)
    X = T_0 * np.cos(a_0) - F_N0 * np.sin(a_0)- D_0 - L_h * np.sin(e)

    #Z = W*np.cos(a_0) - F_N0 - L_wf * np.cos(a_0) - D_0 * np.sin(a_0) - L_h* np.cos(a_0 - e)
    Z = W - T_0 * np.sin(a_0) - F_N0 * np.cos(a_0) - L_wf - L_h* np.cos(e)

    #M = F_N0 * (X_cg - 0) - L_wf * (X_ac_wf - X_cg)*np.cos(a_0) - D_0 * (X_ac_w - X_cg) * np.sin(a_0) - L_h * (X_ac_h - X_cg) * np.cos(a_0
    M = F_N0 * (X_cg - 0) - L_wf * (X_ac_wf - X_cg)*np.cos(a_0) - D_0 * (X_ac_w - X_cg) * np.sin(a_0) - L_h * (X_ac_h - X_cg) * np.cos(a_0 -

    return [X, Z, M]

# Initial guess for the variables
initial_guess = [np.radians(10), np.radians(-5), 2000]  # Provide your initial guess here
radtodeg = 180/np.pi
# Solve the equations
solution = fsolve(equations, initial_guess)

print("Alpha trim is ", solution[0]* radtodeg, '\n')
print("Incidence Angle is ", (solution[1]) * radtodeg, "\n")
print("Trim Thrust is ", solution[2], "\n")

alpha=solution[0]
i_h=solution[1]
thrust=solution[2]
```

```
Alpha trim is  10.628368614640332

Incidence Angle is  -4.909762693703019

Trim Thrust is  1849.2467118310285
```

```
#Most Forward CG
def CL(a,ih):
    return CL_a_wf * a + n_H *CL_a_H * (a * (1-de_da) + ih)*(S_H/S_W)

def CL_H(a, ih):
  return CL_a_H * (a*(1 - de_da) + ih)

def CL_wf(a):
  return CL_a_wf * a

def CM(a,ih):
    return CL_wf(a) * (X_cg - X_ac_wf) - n_H * CL_H(a,ih)* (X_ac_h - X_cg) * (S_H/S_W)


CL_min = W / (q_inf * S_W)

a = np.linspace(-0.2,0.2,1000)
xcg_high = 27
xcg_low = 0
for i in range(27):
    CL_list = []
    CM_list = []
    xcg = (xcg_high+xcg_low)/2
    ilow = -25*np.pi/180
    iup = 15*np.pi/180
    guess_ih = (iup+ilow)/2
    for j in range(len(a)):
      CL_list.append(CL(a[j],guess_ih))
      CM_list.append(abs(CM(a[j],guess_ih)))
    l = CM_list.index(min(CM_list))
    alpha = a[l]*180/np.pi
    CLguess = CL_list[l]
    if CLguess < CL_min:
      iup = guess_ih
      ilow = ilow
    if CLguess > CL_min:
          ilow = guess_ih
          iup = iup
    if abs(guess_ih*180/np.pi) < 20:
        xcghigh = xcg
    if abs(guess_ih*180/np.pi) > 20:
        xcglow = xcg

print('Forward Most CG Location',xcg)
print()
```

```
Forward Most CG Location 13.5
```

Double-click (or enter) to edit

```
#Rest of Table 1
alpha = solution[0]
e = de_da * alpha
AR_w=3.6
M=0.52
beta=math.sqrt(1-M**2)

c_la=7.35593
k=c_la/(2*math.pi)

half_chord=39.69

Normal_force = 2 * q_inf * A_in * np.cos(alpha) ** 2 * np.sin(alpha)
print("The normal force is", Normal_force, "\n")
```

```
C_D0 = 0.0145 + 0.1*((CL_a_W * alpha)**2)
print("C_D0", C_D0, "\n")

D_0 = q_inf * S_W * C_D0

C_L0 = CL_a_wf*alpha + n_H*(S_H/S_W)*CL_a_H*alpha*np.cos(e)
print("C_L0", C_L0, "\n")

prop_CP_M0=(Normal_force*(X_cg-0))/(q_inf*cw_w*S_W)
print("Propulsive force", prop_CP_M0, "\n")

#work=math.sqrt(((3.6**2 * beta**2) /k**2) * (1+((math.tan(math.radians(half_chord))**2)/beta**2))+4)
#CL_alpha=(2*math.pi*3.6)/(2 + work)
CL_alpha = CL_a_wf + CL_a_H * n_H * (S_H / S_W) * (1 - de_da)
print("Big CL_a", CL_alpha, "\n")

deltaX_ach = (X_ac_h - X_cg)/ cw_w
deltaX_acwf = (X_cg - X_ac_wf)/ cw_w

CM_a = CL_a_wf*deltaX_acwf - CL_a_H * n_H * (S_H/S_W) * deltaX_ach * (1- de_da)
print("C_Ma = ", CM_a,"\n")

CM_ih = -CL_a_H*n_H * (S_H/S_W) * deltaX_ach
print("CM_ih = ", CM_ih,"\n")

C_Da=0.2*CL_a_W**2*alpha
print("C_Da", C_Da, "\n")

X_bar_np=(CL_a_wf * (X_ac_wf/cw_w ) + CL_a_H*n_H*(S_H/S_W)*(X_ac_h/cw_w)*(1-de_da))/CL_alpha
print("X_bar_np",X_bar_np, "\n")

SM=-CM_a/CL_alpha
print("SM", SM, "\n")

CM_ih=-CL_a_H*n_H*((S_H/S_W) * deltaX_ach)
print("CM_ih", CM_ih,"\n")

CL_ih=CL_a_H*n_H*(S_H/S_W)
print("CL_ih", CL_ih, "\n")
```

```
The normal force is 499.7177220566599

C_D0 0.0483688188104683

C_L0 0.6572204881969858

Propulsive force 0.03042006007273037

Big CL_a 3.2397062500000002

C_Ma =  -0.40579734195826306

CM_ih =  -0.5278325516441822

C_Da 0.3651624149091098

X_bar_np 2.4018173145785666

SM 0.12525744948581774

CM_ih -0.5278325516441822

CL_ih 0.409625
```

```
# Task 2
import math
import numpy

q_c_W=np.radians(45)
q_c_H=np.radians(45)
q_c_V=np.radians(45)
M=0.52
U_0=503.36 # ft/s
C_Mu=0
I_xx=7210 # slug-ft^2
I_yy=31000 # slug-ft^2
I_zz=37000 # slug-ft^2
I_xz=1000 # slug-ft^2
m=22000/32.2
sos = 968 #speed of sound at cruising altitude (ft/s)

C_DM= ((M*math.cos(q_c_W)**2)/(1 - (M**2) * math.cos(q_c_W)**2)) * C_D0
print("C_DM", C_DM, "\n")

C_LM=((M*math.cos(q_c_W)**2)/(1 - (M**2) *math.cos(q_c_W)**2)) * C_L0
print("C_LM", C_LM, "\n")

C_Mu=0
print("C_Mu", C_Mu, "\n")

C_Mq= -n_H * CL_a_H* (S_H/S_W) * (X_ac_h-X_cg)**2 * (math.cos(alpha)**2) *(1/(U_0*cw_w))
print("C_Mq", C_Mq, "\n")

C_Ma_dot= -n_H * (S_H/S_W) * CL_a_H * (X_ac_h-X_cg) * (X_ac_h-X_ac_w) * (math.cos(alpha)**2) * de_da*(1/(U_0*cw_w))
print("C_Ma_dot", C_Ma_dot,"\n")


C_Du = C_DM / sos
X_u=-((C_Du+(2/U_0)*C_D0)*q_inf*S_W)/m
#X_u = (-q_inf * S_W * (2*C_D0)) / (m*U_0)  #Maybe this equation
print("X_u", X_u, "\n")

C_Lu = C_LM / sos

Z_u=-((C_Lu + (2/U_0) * C_L0)* q_inf * S_W)/m
#Z_u = (-q_inf * S_W * (2*C_L0)) / (m*U_0)
print("Z_u", Z_u, "\n")

M_u=-((C_Mu+(2/U_0)*-prop_CP_M0)*q_inf*S_W*cw_w)/I_yy
print("M_u", M_u, "\n")

X_a=((-C_Da+C_L0)*q_inf*S_W)/m
print("X_a", X_a, "\n")

Z_a=-((CL_alpha+C_D0)*q_inf*S_W)/m
print("Z_a", Z_a, "\n")

M_a=(CM_a*q_inf*S_W*cw_w)/I_yy
print("M_a", M_a, "\n")

M_q=(C_Mq*q_inf*S_W*cw_w)/I_yy
print("M_q", M_q, "\n")

M_a_dot=(C_Ma_dot*q_inf*S_W*cw_w)/I_yy
print("M_a_dot", M_a_dot, "\n")
```

```
    C_DM 0.014541966802407215

    C_LM 0.19759172864386718

    C_Mu 0

    C_Mq -0.015480364668898882

    C_Ma_dot -0.010373465043586445

    X_u -0.01134175628262406

    Z_u -0.15410826198352734

    M_u 0.001729330959888878
```

```
     X_a 15.98623401381166
```

```
# Task 3
import numpy as np
import math
from numpy.linalg import eig

A_mat=np.array([[X_u, X_a, 0, -32.2],
                [(Z_u / U_0), (Z_a / U_0), 1, 0],
                [M_u + (M_a_dot * Z_u) / U_0, M_a + (M_a_dot * Z_a) / U_0, (M_q + M_a_dot), 0],
                [0, 0, 1, 0]])
```