italicized text# Normal *italicized text*

```
######################################## 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 - e

    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
while X_cg != 0.0:
   # print(f'i_h = {fsolve(equations,initial_guess)[1]*radtodeg}')
   if abs(abs(fsolve(equations,initial_guess)[1]*radtodeg) - 20) < 0.1:
      print(f'forward most xcg = {X_cg}')
      break
   X_cg -= .0001
X_cg = 27
```

```
forward most xcg = 24.323900000006237
```

Task 1 of project ✍

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

SM=X_bar_np - X_cg#-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 28.485553350901796

SM 1.485553350901796

CM_ih -0.5278325516441822
```

CL_ih 0.409625

Table 2 ;0

```
# 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
print("C_Du", C_Du, "\n")
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
print("C_Lu", C_Lu, "\n")

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=((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

C_Du 1.5022692977693404e-05

X_u -0.01134175628262406

C_Lu 0.00020412368661556528

Z_u -0.15410826198352734

M_u -0.001729330959888878

X_a 15.98623401381166

Z_a -179.97769044094454

M_a -5.805985576082417

M_q -0.2214868474652724

M_a_dot -0.14841937634784372

Table 3 ✴

```
# 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]])

def values(eig):
    A = eig.real
    B = eig.imag
    natural_freq = np.sqrt(A**2 + B**2)
    damping_ratio = -A / natural_freq
    time_to_half = np.log(2) / np.abs(A)
    time_constant = 1/ np.abs(A)
    cycle_to_half = abs(time_to_half / (2*np.pi / B))

    return     print("Real", A, "\nImaginary", B, '\nDamping Ratio =', damping_ratio , "\n Natural Frequency =", natural_freq,
            "\n Time to half", time_to_half,"\n Time Constant =", time_constant,
            "\n Cycles-to-half =", cycle_to_half)

w,v=eig(A_mat)
print("Exact Eigen values \n", w)
print("")
print("Exact Eigen vector \n", v)
print("")
print("Phugoid ", values(w[3]))
print('')
print("Short Period", values(w[1]))
```

```
Exact Eigen values
 [-0.36629098+2.39859519j -0.36629098-2.39859519j -0.00310932+0.07956431j
 -0.00310932-0.07956431j]

Exact Eigen vector
 [[ 9.23303364e-01+0.00000000e+00j  9.23303364e-01-0.00000000e+00j
   9.99996796e-01+0.00000000e+00j  9.99996796e-01-0.00000000e+00j]
 [ 5.89619121e-02-1.24817632e-01j  5.89619121e-02+1.24817632e-01j
  -3.05661653e-04-1.16034432e-06j -3.05661653e-04+1.16034432e-06j]
 [ 2.99154419e-01+1.42516459e-01j  2.99154419e-01-1.42516459e-01j
   1.97910749e-04-2.47310359e-05j  1.97910749e-04+2.47310359e-05j]
 [ 3.94504577e-02-1.30745183e-01j  3.94504577e-02+1.30745183e-01j
  -4.07415745e-04-2.47150957e-03j -4.07415745e-04+2.47150957e-03j]]

Real -0.003109321640247104
Imaginary -0.07956431356678743
Damping Ratio = 0.039049543015698296
 Natural Frequency = 0.07962504552222613
 Time to half 222.92553191919365
 Time Constant = 321.613559387355
```

```
 Cycles-to-half = 2.8229180036110377
Phugoid  None

Real -0.3662909820210849
Imaginary -2.3985951879302982
Damping Ratio = 0.15096053406814078
 Natural Frequency = 2.4264022665403924
 Time to half 1.8923402829503606
 Time Constant = 2.7300699418869034
 Cycles-to-half = 0.7223976493936727
Short Period None
```

Table 4 ✈

```
###### Table 4 ########## added in portions
g=32.2

na=-(Z_a/g)
print("n/alpha is ", na, "\n")


CAP=(2.3965347938016235**2)/na
print("CAP is", CAP, "\n")
```

```
n/alpha is  5.589369268352314

CAP is 1.0275540480786443
```

Table 5 🧦

```
###### Table 5 ##########
# Lateral-Directional Stability Coefficients (Stability Axis System)
# b=beta

AR_V=1.5
do_db=0.12
S_V=70
n_V=1
chord_half_V=math.radians(34.64)
k=1

C_YbWF=-(2*A_in)/S_W
print("C_YbWF is", C_YbWF,"\n")

AR_Veff=1.55*AR_V
print("AR_Veff is", AR_Veff, "\n")

C_LaV=(2*math.pi*AR_Veff)/(2+math.sqrt(((AR_Veff**2*beta**2)/k**2)*(1+math.tan(chord_half_V)**2/beta**2)+4))
print("C_LaV is", C_LaV, "\n")

C_YbV=-k*C_LaV*(1+do_db)*n_V*(S_V/S_W)
print("C_YbV is", C_YbV, "\n")

C_Yb = C_YbWF + C_YbV
print("********C_Yb is**********", C_Yb, "\n")

# C_Lbeta
chord_LEW=math.radians(49.47)

b_w=37.947
lam_w=0.24085
b_h=18.439
lam_h=0.084647
Y_sq_ACW=((b_w**2)/24)*((1+3*lam_w)/((1+lam_w)))
Y_sq_ACH=((b_h**2)/24)*((1+3*lam_h)/((1+lam_h)))
print("Y_sqiggle_ACW", Y_sq_ACW, "\n")
print("Y_sqiggle_ACH", Y_sq_ACH, "\n")

Y_ACW=(b_w/2)*((1+2*lam_w)/(3*(1+lam_w)))
print("Y_ACW", Y_ACW, "\n")

Y_ACH=(b_h/2)*((1+2*lam_h)/(3*(1+lam_h)))
print("Y_ACH", Y_ACH, "\n")

X_LEmac=2.8362
X_APPV=35
X_ACV=X_APPV + X_LEmac + 0.32*7.32
Z_cg=0
Z_ACV=0
delta_Z_ACVS=-(X_ACV - X_cg)*math.sin(alpha)
print("Delta_Z_ACVs", delta_Z_ACVS, "\n")

delta_X_ACVs=(X_ACV - X_cg)*math.cos(alpha)
print("Delta_X_ACVs", delta_X_ACVs, "\n")

C_YP=C_YbV*(delta_Z_ACVS/U_0)
print("*************C_YP***************", C_YP, "\n")

C_Yr=-C_YbV*(delta_X_ACVs/U_0)
print("**************C_Yr**************", C_Yr, "\n")

C_Nr=C_YbV*((delta_X_ACVs**2)/(b_w*U_0))
print("**************C_Nr**************", C_Nr, "\n")

vol=1000
C_NbWF=-1.3*(vol/(S_W*b_w))-(2*A_in*X_cg)/(S_W*b_w)
print("C_NbWF", C_NbWF, "\n")

C_NbV=-C_YbV*(delta_X_ACVs/b_w)
print("C_NbV", C_NbV, "\n")

C_Nb=C_NbWF + C_NbV
print("**************C_Nb**************", C_Nb, "\n")

C_LW = CL_a_W * alpha
print("C_LW", C_LW, "\n")
```

```
C_LbWF = -C_LW*math.sin(2*chord_LEW)*(Y_ACW/cw_w)
print("C_LbWF", C_LbWF,"\n")

C_LbV = C_YbV*(delta_Z_ACVS/b_w)
print("C_LbV", C_LbV, "\n")

alpha_H = alpha + i_h - (de_da*alpha)
print("Alpha H", np.degrees(alpha_H), "\n")

C_LH=CL_a_H*alpha_H
print("C_LH", C_LH, "\n")

C_LbH = -n_H*(S_H/S_W)*(C_LH*math.sin(2*chord_LEW))*(Y_ACH/cw_h)
print("C_LbH", C_LbH, "\n")

C_Lb = C_LbWF + C_LbV + C_LbH
print("**************C_Lb**************", C_Lb, "\n")


C_LPWF = -((CL_a_W*Y_sq_ACW)/(b_w*U_0))
print("C_LPWF", C_LPWF,"\n")


C_LPV = -n_H*(S_H/(S_W*b_w*U_0))*CL_a_H*Y_sq_ACH
print("C_LPV", C_LPV, "\n")

C_LP = C_LPWF + C_LPV
print("**************C_LP**************", C_LP, "\n")

C_NPWF = -(C_LW * Y_sq_ACW)/(b_w*U_0)
print("C_NPWF", C_NPWF, "\n")

C_NPH = -C_YbV*((delta_X_ACVs*delta_Z_ACVS)/(b_w*U_0))
print("C_NPH", C_NPH, "\n")

C_NP = C_NPWF + C_NPH
print("**************C_NP**************", C_NP, "\n")

C_LrWF = (2*C_LW*Y_sq_ACW)/(b_w*U_0)
print("C_LrWF", C_LrWF, "\n")

C_LrV = -C_YbV*((delta_X_ACVs*delta_Z_ACVS)/(b_w*U_0))
print("C_LrV", C_LrV, "\n")

C_Lr = C_LrWF + C_LrV
print("**************C_Lr**************", C_Lr, "\n")
```

```
     ********C_Yb is********** -0.6209943855396328

     Y_sqiggle_ACW 83.29064106737823

     Y_sqiggle_ACH 16.377672041666592

     Y_ACW 7.552090623363019

     Y_ACH 3.3129998463401766

     Delta_Z_ACVs -2.4306346877779736

     Delta_X_ACVs 12.95250991022861

     *************C_YP************** 0.0026365084488728786

     **************C_Yr************** 0.014049582186966137

     **************C_Nr************** -0.004795566250592938

     C_NbWF -0.1390096713837721

     C_NbV 0.18636513267534388
```

```
Alpha H -2.252670540042936
C_LH -0.12884022911382453
C_LbH 0.007855107122223752
**************C_Lb************** -0.3232506090584371
C_LPWF -0.013680315269068656
C_LPV -0.0003512226184870033
**************C_LP************** -0.01403153788755566
C_NPWF -0.00253769884413485
C_NPH -0.0008999236253834932
**************C_NP************** -0.003437622469518343
C_LrWF 0.0050753976882697
C_LrV -0.0008999236253834932
**************C_Lr************** 0.004175474062886207
```

Table 7 ;D

```
# Table 6
# b=beta

Y_b=(C_Yb*q_inf*S_W)/m
print("**************Y_b**************", Y_b, "\n")

Y_P=(C_YP*q_inf*S_W)/m
print("**************Y_P**************", Y_P, "\n")

Y_r=(C_Yr*q_inf*S_W)/m
print("**************Y_r**************", Y_r, "\n")



I_mat = np.array([[7210, 0, -1000],
                  [0, 31000, 0],
                  [-1000, 0, 37000]])

b_to_s = np.array([[np.cos(alpha), 0, np.sin(alpha)],
                   [0, 1, 0],
                   [-np.sin(alpha), 0, np.cos(alpha)]])
I_mat_stab = b_to_s.dot(I_mat.dot(np.transpose(b_to_s)))

print(I_mat_stab)
Is_xx = I_mat_stab[0, 0]
Is_yy = I_mat_stab[1, 1]
Is_zz = I_mat_stab[2, 2]
Is_xz = I_mat_stab[0, 2] * (-1)
```