```python
import numpy as np
import matplotlib.pyplot as plt


def wing_weight(W_TO, n_ult, A, del_quar, S, lam, tcm, V_H, W_F, q):
    wing_weight_USAF = 
96.948*((W_TO*n_ult/(10**5))**(.65)*(A/np.cos(del_quar))**(.57)*(S/100)**(.61)*((1+lam)/2*(tcm))**(.36)*(1+V_H/500)**(.5))**(.993)

    wing_weight_ces = .04674*(W_TO)**(.397)*(S**.36)*n_ult**.397*A**1.712
    #wing_weight_raymer = .036*S**.758*(.5*W_F)**.0035*(A/((np.cos(del_quar))**2))*q**.006*lam**.04*(100*tcm/np.cos(del_quar))**(-
0.3)*(n_ult*W_TO)**.49
    return (wing_weight_USAF + wing_weight_ces) / 2

def empennage_weight(W_TO, n_ult, S_v, b_v, trv, A_v, del_quar_v):
    W_V_USAF = 98.5*((W_TO*n_ult/(10**5))**(.87)*(S_v/100)**(1.2)*.289*(b_v/trv)**(.5))**(.458)
    W_V_tor = .04*(n_ult*S_v**2)**.75
    empennage_weight = (W_V_USAF + W_V_tor )/2
    return empennage_weight

def fuselage_weight(W_TO, n_ult, l_f, w_f, h_f, V_C, N_pax, S_f, l_t, q):
    fuselage_weight_USAF = 200*((W_TO*n_ult/(10**5))**(.286)*(l_f/10)**(.857)*((w_f+h_f)/10)*(V_C/100)**(.338))**(1.1)
    fuselage_weight_ces = .04682*W_TO**.692*N_pax*.374*l_f**.59/100
    fuselage_weight_raymer = .052*S_f**1.086*(n_ult*W_TO)**.177*l_t**(-0.051)*(l_f/w_f)**(-.072)*q**.241
    return (fuselage_weight_USAF + fuselage_weight_ces + fuselage_weight_raymer) / 3


def engine_weight(P_TO, N_e, N_p, N_bl, D_p, k_p=.55):
    #includes nacelle, air induction, propeller, propulsion system
    W_eng = k_p*P_TO
    engine_weight_USAF = 2.575*W_eng**(.992)*N_e
    W_n = .37*P_TO
    W_prop = 31.92*N_p*N_bl**.391*((D_p)*P_TO/N_e/1000)**.782
    W_p = 65
    engine_weight_ces = W_eng + W_n + W_prop + W_p
    return (engine_weight_USAF + engine_weight_ces)/2

def fuel_system_weight(W_F, K_fsp, fracfuel, N_t, N_e):
    W_fs_USAF = 2.49*((W_F/K_fsp)**(.6)*(1/(1+fracfuel))**(.3)*(N_t)**(.2)*N_e**(.13))**(1.21)
    W_fs_tor = 2*(W_F/5.87)**.667
    return (W_fs_USAF + W_fs_tor)/2

def flight_control_sys_weight(W_TO):
    W_fc_USAF = 1.066*(W_TO)**(.626)
    W_fc_tor = .23*W_TO**(2/3)
    return (W_fc_USAF + W_fc_tor)/2

def electrical_system_weight(W_fs, W_iae, W_TO):
    W_els_USAF = 426*((W_fs + W_iae)/1000)**(.51)
    W_els_ces = .0268*(W_TO)
    return (W_els_USAF + W_els_ces)/2

def avionics_weight(N_pax):
    W_iae = 33*N_pax
    return W_iae

def AC_weight(W_TO, N_pax, W_iae, M_D):
    W_api = .265*(W_TO)**(.52)*(N_pax)**(.68)*(W_iae)**(.17)*M_D**(.08)
    return W_api

def APU_weight(W_TO):
    W_APU = .013*W_TO
    return W_APU

def furnishings_weight(N_pax, W_TO):
    W_fur = .412*N_pax**(1.145)*W_TO**(.489)
    return W_fur


def total_weight_calc():
    W_TO = 10910
    #payload weight
    W_payload = 4000
    #passenger weight
    W_pax = 190
    #landing gear weight
    W_lgm = 249
    W_lgf = 67
    #boom weight
    W_boom = 150
    #mission fuel weight [lbs]
    W_F = 693.36
    #wing aspect ratio
    A = 10
    #wing quarter-chord sweep angle
    del_quar = np.deg2rad(5)
    #wing area [ft^2]
    S = 193.6
    #cruise dynamic pressure
    q=(1/2*.002048 * 197.5**2)
    #wing taper ratio
    lam = 0.7
    #maximum wing thickness ratio
```

```
    tcm = 0.18
    #maximum level speed at sealevel [kts]
    V_H = 394/309*117
    #vertical tail area [ft]
    S_v = 40.4624
    #vertical tail span [ft]
    b_v = 8.9
    #distance from the wing quarter chord to tail quarter chord
    l_t = 21.4
    #vertical tail maximum root thickness [ft]
    trv = 0.16
    #vertical tail aspect ratio
    A_v = 1.88
    #vertical tail quarter chord sweep angle
    del_quar_v = np.deg2rad(15)
    #fuselage length [ft]
    l_f = 35.5
    #max fuselage width [ft]
    w_f = 4.4
    #max fuselage height [ft]
    h_f = 6.5
    #fuselage wetted area
    S_f = 426.5
    #design cruise speed [KEAS]
    V_C = 128
    #engine takeoff power [shp]
    P_TO = 620
    #number of engine
    N_e = 1
    #number of propeller
    N_p = 1
    #number of propeller blades
    N_bl = 5
    #diameter of propeller
    D_p = 9
    #constant
    K_fsp = 5.87
    #fuel fraction of tanks which are integral
    fracfuel = 1
    #number of separate engine tanks
    N_t = 1
    #number of passengers
    N_pax = 1
    #design dive mach number
    M_D = .438
    tolerance = .05
    iter = 0
    W_TO_new = 0
    W_TO_old = W_TO
    while abs(W_TO_new-W_TO_old)/W_TO_old >= .05:
        W_TO_old = W_TO
        #positive design limit load factor
        n_lim_pos = 2.1 + 24000/(W_TO + 10000)
        #design ultimate load factor
        n_ult = 1.5*n_lim_pos
        #design landing weight [lbs]
        W_L = W_TO-W_payload
        W_w = wing_weight(W_TO, n_ult, A, del_quar, S, lam, tcm, V_H, W_F, q)
        W_em = empennage_weight(W_TO, n_ult, S_v, b_v, trv, A_v, del_quar_v)
        W_fus = fuselage_weight(W_TO, n_ult, l_f, w_f, h_f, V_C, N_pax, S_f, l_t, q)
        W_eng = engine_weight(P_TO, N_e, N_p, N_bl, D_p, k_p=.55)
        W_fs = fuel_system_weight(W_F, K_fsp, fracfuel, N_t, N_e)
        W_fc = flight_control_sys_weight(W_TO)
        W_iae = avionics_weight(N_pax)
        W_els = electrical_system_weight(W_fs, W_iae, W_TO)
        W_api = AC_weight(W_TO, N_pax, W_iae, M_D)
        W_APU = APU_weight(W_TO)
        W_fur = furnishings_weight(N_pax, W_TO)
        W_winglets = 1/12.7*2*W_w
        W_hopper = 64.125*1.6
        W_tfo = 44

        W_TO_new = W_payload + W_F + W_pax + W_boom + 2*W_w + W_em + W_fus + W_lgm + W_lgf + W_eng + W_fs + W_fc + W_iae + W_els
+ W_api + W_APU + W_fur + W_winglets + W_hopper + W_tfo
        W_TO = W_TO_new
        iter = iter + 1

    return iter, W_payload, W_F, W_pax, n_lim_pos, W_w, W_em, W_fus, W_lgm, W_lgf, W_eng, W_fs, W_fc, W_iae, W_els, W_api, W_APU,
W_fur, W_winglets, W_hopper, W_tfo, W_boom, W_TO_new


iter, W_payload, W_F, W_pax, n_lim_pos, W_w, W_em, W_fus, W_lgm, W_lgf, W_eng, W_fs, W_fc, W_iae, W_els, W_api, W_APU, W_fur,
W_winglets, W_hopper, W_tfo, W_boom, W_TO_new = total_weight_calc()
print("Iterations: ", iter)
print("Payload: ", W_payload)
print("Fuel: ", W_F)
print("Passenger: ", W_pax)
print("Positive Load Limit:", n_lim_pos)
print("Wings: ", 2*W_w)
print("Winglets: ", W_winglets)
print("Empennage: ", W_em)
print("Fuselage: ", W_fus)
```

```python
print("Landing Gear: ", W_lgm, W_lgf)
print("Engine: ", W_eng)
print("Fuel Systems: ", W_fs)
print("Flight Control Systems: ", W_fc)
print("Avionics: ", W_iae)
print("Electrical Systems: ", W_els)
print("AC, Pressure, Anti-Ice Systems: ", W_api)
print("APU: ", W_APU)
print("Furnishings: ", W_fur)
print("Hopper: ", W_hopper)
print("Trapped Fuel and Oil: ", W_tfo)
print("Boom: ", W_boom)
print("Gross Takeoff Weight: ", W_TO_new)



def cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F, W_pax, W_hopper, W_payload, W_boom, W_fc, W_els,
W_iae, W_api, W_APU, W_fur):
    x_cg = (12.79*W_w + 33.69*W_w + 31.314*W_em + 22.142*W_winglets + 12.425*W_fus + 31.695*W_eng + 3.45*W_lgf + 21.725*W_lgm +
22.8*W_fs + 10.364*W_F + 7.554*W_pax + 14.65*W_hopper + 14.65*W_payload + 13.838*W_boom + 22.142*W_fc + 3.434*W_els + 4.696*W_iae
+ 7.008*W_api + 2.485*W_APU + 6.717*W_fur)/( 2*W_w + W_em + W_winglets + W_fus + W_eng + W_lgf + W_lgm + W_fs + W_F + W_pax +
W_hopper + W_payload + W_boom + W_fc + W_els + W_iae + W_api + W_APU + W_fur)
    return x_cg

def cg_location_gear_down(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F, W_pax, W_hopper, W_payload, W_boom, W_fc,
W_els, W_iae, W_api, W_APU, W_fur):
    x_cg = (12.79*W_w + 33.69*W_w + 31.314*W_em + 22.142*W_winglets + 12.425*W_fus + 31.695*W_eng + 2.155*W_lgf + 21.725*W_lgm +
22.8*W_fs + 10.364*W_F + 7.554*W_pax + 14.65*W_hopper + 14.65*W_payload + 13.838*W_boom + 22.142*W_fc + 3.434*W_els + 4.696*W_iae
+ 7.008*W_api + 2.485*W_APU + 6.717*W_fur)/( 2*W_w + W_em + W_winglets + W_fus + W_eng + W_lgf + W_lgm + W_fs + W_F + W_pax +
W_hopper + W_payload + W_boom + W_fc + W_els + W_iae + W_api + W_APU + W_fur)
    return x_cg

x_cg = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F, W_pax, W_hopper, W_payload, W_boom, W_fc, W_els,
W_iae, W_api, W_APU, W_fur)
print("Center of Gravity: ", x_cg)

x_cg_empty = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, 0, W_pax, W_hopper, 0, W_boom, W_fc, W_els,
W_iae, W_api, W_APU, W_fur)
print("Empty Center of Gravity: ", x_cg_empty)


w1 = 6.039
w2 = 8.052
w3 = 51.658
w4 = 344.792
w5 = 49.752
w6 = 119.864
cg_gear_down_TO = cg_location_gear_down(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1, W_pax, W_hopper,
W_payload, W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_takeoff_start = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1, W_pax, W_hopper, W_payload,
W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_cruise_start = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1-w2, W_pax, W_hopper, W_payload,
W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_spray_start = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1-w2-w3, W_pax, W_hopper, W_payload,
W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_cruise_back_start = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1-w2-w3-w4, W_pax, W_hopper,
W_payload-4000, W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_loiter_start = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1-w2-w3-w4-w5, W_pax, W_hopper,
W_payload-4000, W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_land_start = cg_location(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1-w2-w3-w4-w5-w6, W_pax, W_hopper,
W_payload-4000, W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)
cg_gear_down_land = cg_location_gear_down(W_w, W_em, W_winglets, W_fus, W_eng, W_lgf, W_lgm, W_fs, W_F-w1-w2-w3-w4-w5-w6, W_pax,
W_hopper-4000, W_payload, W_boom, W_fc, W_els, W_iae, W_api, W_APU, W_fur)

cg_envelope = [cg_gear_down_TO, cg_takeoff_start, cg_cruise_start, cg_spray_start, cg_cruise_back_start, cg_loiter_start,
cg_land_start, cg_gear_down_land]
print(cg_envelope)

cg_location_envelope = np.zeros_like(cg_envelope)
for i in range(len(cg_envelope)):
    cg_location_envelope[i] = cg_envelope[i]/4.45
print(cg_location_envelope)
cg_weights = [W_TO_new-w1, W_TO_new-w1, W_TO_new-w1-w2, W_TO_new-w1-w2-w3, W_TO_new-w1-w2-w3-w4-4000,
W_TO_new-w1-w2-w3-w4-w5-4000, W_TO_new-w1-w2-w3-w4-w5-w6-4000, W_TO_new-w1-w2-w3-w4-w5-w6-4000]
print(cg_weights)

plt.figure()
plt.plot(cg_location_envelope, cg_weights)

plt.axvline(x=16.9/4.45, linestyle='--')
plt.axvline(x=18.45/4.45, linestyle='--')

plt.scatter(cg_location_envelope[0], cg_weights[0], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[1], cg_weights[1], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[2], cg_weights[2], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[3], cg_weights[3], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[4], cg_weights[4], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[5], cg_weights[5], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[6], cg_weights[6], color='black', s=10, zorder=2)
plt.scatter(cg_location_envelope[7], cg_weights[7], color='black', s=10, zorder=2)

plt.title("C.G. Location for Full Length Mission")
```

```
plt.xlabel("C.G. Location as a fraction of MAC $(\\frac{x_{c.g.}}{\\bar{c}})$")
plt.ylabel("Gross Weight ($W$) [lbs]")



plt.show()
```