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

#!!! note that all power loading values are in lbf/hp
#wing loading vector [lbm/ft^2]
wing_loading = np.linspace(0, 80, 200)
#stall velocity at takeoff [ft/s] (145) (120)
v_stall_TO = 120
#stall velocity during cruise [ft/s]
v_stall_flight = 145
#overall maximum coefficient of lift
cl_max = 2.0
#maximum coefficient of lift at cruise
cl_max_cruise = 1.6
#Takeoff distance [ft]
s_TO = 1000
#density on the ground (should be at sea level for Davis) [slug/ft^3]
rho_ground = .002378
#density at sea level [slug/ft^3]
rho_sea = .002378
#density at cruise [slug/ft^3]
rho_cruise = .002048
#maximum coefficient of lift at takeoff
cl_max_TO = 1.7333
#total landing distance [ft]
s_L = 1500
#obstacle clearance distance [ft]
s_a = 500
#maximum coefficient of lift at landing conditions
cl_max_land = 2.0
#span efficiency factor
e = 0.97
#Aspect ratio
AR = 5
#stall factor (in FAR 23)
k_s = 1.3
#drag coefficient at zero lift
cd_0 = .02862
#max coefficient of lift at climb
cl_max_climb = 1.7333
#cruise weight [lbm]
W_cruise = 12214
#takeoff weight [lbm]
W_takeoff = 12214
#landing weight [lbm] !!!
W_landing = 12214-4000
#cruise speed [ft/s] potentially 267
v_cruise = 212
#propeller efficiency
eta_p = 0.9
#dynamic pressure at cruise [lbf/ft^2] slugs/ft^3 * ft^2/s^2 = slugs/ft/s^2 = lbf/ft^2
q_cruise = (1/2*rho_cruise * v_cruise**2)
#power at cruise [hp]
P_cruise = 787.5
#power at takeoff [hp]
P_takeoff = 950
#turn radius [ft] !!!
R_turn = 1800


#Stall Constraint (plot on x axis as vertical line)
def stall_constraint(v_stall, rho_cruise, cl_max):
    wing_loading_stall = 1/2 * rho_cruise * v_stall**2 * cl_max
    return wing_loading_stall

#Takeoff constraint (function of W/S) !POWER CORRECTION
def takeoff_constraint(s_TO, rho_ground, rho_sea, cl_max_TO, wing_loading, eta_p):
```

```python
    a = .0149
    b = 8.314
    c = -s_TO
    coefficients = [a, b, c]
    solutions = np.roots(coefficients)
    TOP_23 = max(solutions)
    takeoff_constraint = wing_loading / (rho_ground/rho_sea * cl_max_TO * TOP_23)
    V = np.sqrt(2 / rho_ground / cl_max_TO * wing_loading)
    takeoff_constraint_power = 550*eta_p/V * 1/takeoff_constraint
    return takeoff_constraint, takeoff_constraint_power

#Landing constraint (plot on x axis as multiple vertical lines for each cl)
def landing_constraint(s_L, s_a, cl_max_land, rho_ground, rho_sea, W_landing, W_takeoff):
    wing_loading_landing = (s_L - s_a) / 80 * cl_max_land * rho_ground/rho_sea
    wing_loading_landing_corrected = wing_loading_landing / (W_landing / W_takeoff)
    return wing_loading_landing_corrected

#Climb constraint (plot on y axis as horizontal line) !POWER CORRECTION
def climb_constraint(wing_loading, e, AR, k_s, cd_0, cl_max_climb, rho, eta_p, W, W_takeoff,
G):
    k = 1 / (np.pi* e * AR)
    climb_constraint = k_s**2 * cd_0 / cl_max_climb + k * cl_max_climb / k_s**2 + G
    climb_constraint_corrected = 1/.8 * 1/.94 * W / W_takeoff * climb_constraint
    wing_loading_climb = wing_loading * (W / W_takeoff)
    V = np.sqrt(2/rho / cl_max_climb * wing_loading_climb)
    climb_constraint_corrected_power = 550*eta_p/V * 1/climb_constraint_corrected
    return climb_constraint_corrected, climb_constraint_corrected_power

#Cruise constraint (function of W/S) !POWER CORRECTION
def cruise_constraint(e, AR, W_cruise, W_takeoff, wing_loading, v_cruise, eta_p, q_cruise,
cd_0, P_cruise, P_takeoff):
    k = 1 / (np.pi* e * AR)
    wing_loading_cruise = wing_loading * (W_cruise / W_takeoff)
    cruise_constraint =  (q_cruise * cd_0 / (wing_loading_cruise) + k / q_cruise *
wing_loading_cruise)
    cruise_constraint_power =  (v_cruise / eta_p / 550 * (q_cruise * cd_0 /
wing_loading_cruise + k / q_cruise * wing_loading_cruise))
    cruise_constraint_corrected = (W_cruise / W_takeoff / (P_cruise / P_takeoff) *
cruise_constraint)
    cruise_constraint_corrected_power = 1 / ((W_cruise / W_takeoff) / (P_cruise / P_takeoff) *
cruise_constraint_power)
    return cruise_constraint_corrected, cruise_constraint_corrected_power

#Ceiling constraint (plot on y axis as horizontal line) !POWER CORRECTION
def ceiling_constraint(e, AR, cd_0, rho_cruise, cl_max_cruise, wing_loading, eta_p, W_cruise,
W_takeoff, P_cruise, P_takeoff):
    k = 1 / (np.pi* e * AR)
    ceiling_constraint = 2*np.sqrt(k*cd_0)
    ceiling_constraint_corrected = ceiling_constraint * W_cruise / W_takeoff / (P_cruise /
P_takeoff)
    wing_loading_ceiling = wing_loading * (W_cruise/W_takeoff)
    V = np.sqrt(2/rho_cruise/cl_max_cruise * wing_loading_ceiling)
    ceiling_constraint_power = 550*eta_p/V * 1/ceiling_constraint_corrected
    return ceiling_constraint, ceiling_constraint_power

#Manuever constraint (function of W/S) !POWER CORRECTION
def maneuver_constraint(e, AR, v_stall_flight, R_turn, W_cruise, W_takeoff, q_cruise, cd_0,
wing_loading, rho_ground, cl_max, eta_p, P_cruise, P_takeoff):
    k = 1 / (np.pi* e * AR)
    g = 32.2
    n = np.sqrt((v_stall_flight**2 / R_turn / g)**2 + 1)
    wing_loading_maneuver = wing_loading * (W_cruise/W_takeoff)
    maneuver_constraint = q_cruise*cd_0/wing_loading + k * n**2 / q_cruise *
wing_loading_maneuver
    V = np.sqrt(2/rho_ground / cl_max * wing_loading_maneuver)
    maneuver_constraint_power = 1/((V / eta_p / 550 * maneuver_constraint) * (W_cruise /
W_takeoff) / (P_cruise / P_takeoff))
```

```python
    return maneuver_constraint, maneuver_constraint_power, n

#Maneuver stall Constraint (plot on x axis as vertical line)
def stall_constraint_maneuver(rho_cruise, v_stall_flight, cl_max, n, W_cruise, W_takeoff):
    wing_loading_stall_maneuver = 1/2 * rho_cruise * (v_stall_flight/np.sqrt(n))**2 * cl_max
    wing_loading_stall_maneuver_corrected = wing_loading_stall_maneuver / (W_cruise /
W_takeoff)
    return wing_loading_stall_maneuver_corrected

wing_loading_stall = stall_constraint(v_stall_flight, rho_cruise, cl_max_cruise)
takeoff_constraint, takeoff_constraint_power = takeoff_constraint(s_TO, rho_ground, rho_sea,
cl_max_TO, wing_loading, eta_p)
wing_loading_landing_corrected = landing_constraint(s_L, s_a, cl_max_land, rho_ground,
rho_sea, W_landing, W_takeoff)
climb_constraint_corrected, climb_constraint_corrected_power = climb_constraint(wing_loading,
1.03, AR, k_s, .0436, cl_max_climb, rho_ground, eta_p, W_takeoff, W_takeoff, 0.04)
climb_constraint_corrected2, climb_constraint_corrected_power2 =
climb_constraint(wing_loading, .97, AR, k_s, .1136, cl_max_land, rho_ground, eta_p, W_landing,
W_takeoff, 0.03)
cruise_constraint_corrected, cruise_constraint_corrected_power = cruise_constraint(1.10, AR,
W_cruise, W_takeoff, wing_loading, v_cruise, eta_p, q_cruise, cd_0, P_cruise, P_takeoff)
ceiling_constraint, ceiling_constraint_power = ceiling_constraint(1.10, AR, cd_0, rho_cruise,
cl_max_cruise, wing_loading, eta_p, W_cruise, W_takeoff, P_cruise, P_takeoff)
maneuver_constraint, maneuver_constraint_power, n = maneuver_constraint(1.10, AR,
v_stall_flight, R_turn, W_cruise, W_takeoff, q_cruise, cd_0, wing_loading, rho_ground,
cl_max_cruise, eta_p, P_cruise, P_takeoff)
wing_loading_stall_maneuver = stall_constraint_maneuver(rho_cruise, v_stall_flight,
cl_max_cruise, n, W_cruise, W_takeoff)

plt.figure()
plt.plot(wing_loading, takeoff_constraint_power * .77, label='Takeoff')
plt.plot(wing_loading, climb_constraint_corrected_power*.77, label='Takeoff Climb')
plt.plot(wing_loading, climb_constraint_corrected_power2*.77, label='Balked Climb')
plt.plot(wing_loading, cruise_constraint_corrected_power*.77, label='Cruise')
plt.plot(wing_loading, ceiling_constraint_power*.77, label='Ceiling')
plt.plot(wing_loading, maneuver_constraint_power*.77, label='Maneuver')
plt.axvline(x=wing_loading_stall_maneuver, color = 'y', linestyle = '-', label='Maneuver
Stall')
plt.axvline(x=wing_loading_stall, color='m', linestyle='-', label='Stall')
plt.axvline(x=wing_loading_landing_corrected, color='c', linestyle='-', label='Landing')
plt.axis([0, 60, 0, 50])
plt.legend(loc=(.68, .45))
plt.grid()
'''x_range = (wing_loading >= 22.2) & (wing_loading <= 31.375)
plt.fill_between(wing_loading, 0, cruise_constraint_corrected_power*.77,
where=wing_loading<=23, color='skyblue')
plt.fill_between(wing_loading[x_range], 0, climb_constraint_corrected_power2[x_range]*.77,
color='skyblue')'''
plt.scatter(32.4, 15.7, color='red',s=70,zorder=2)
plt.scatter(25.4, 17.8, color='red',s=70,zorder=2)
plt.title("Power Loading Specific (W/P$_{\mathrm{Specific}}$) vs. Wing Loading (W/S)")
plt.xlabel("W/S [lbs/ft\u00b2]")
plt.ylabel("W/P$_{\mathrm{Specific}}$ [lbs/hp]")
plt.show()
```