```
import numpy as np
import matplotlib.pyplot as plt
import Constraint_Graph as cg
import a4_weight_estimation as we

x = 25.4
y = 17.8

def takeoff_constraint_converge(x, y, s_TO, rho_ground, rho_sea, cl_max_TO, eta_p):
    S = np.linspace(200, 600, 1000)
    P = np.zeros_like(S)
    for i in range(len(S)):
        S_0 = S[i]
        P[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S_0, P[i], S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading = W/S_0
            takeoff_constraint, takeoff_constraint_power = cg.takeoff_constraint(s_TO,
rho_ground, rho_sea, cl_max_TO, wing_loading, eta_p)
            power_loading_new = takeoff_constraint_power
            P_new = 1/power_loading_new * W
            if abs(P_new - P[i]) <= tolerance:
                converged = True
            P[i] = P_new
    return S, P


def climb_constraint_converge(x, y, e, AR, k_s, cd_0, cl_max_climb, rho, eta_p, G, type):
    S = np.linspace(200, 600, 1000)
    P = np.zeros_like(S)
    for i in range(len(S)):
        S_0 = S[i]
        P[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S_0, P[i], S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading = W/S_0
            if type == 'takeoff':
                climb_constraint, climb_constraint_power = cg.climb_constraint(wing_loading,
e, AR, k_s, cd_0, cl_max_climb, rho, eta_p, W, W, G)
            if type == 'balked':
                climb_constraint, climb_constraint_power = cg.climb_constraint(wing_loading,
e, AR, k_s, cd_0, cl_max_climb, rho, eta_p, W - 4000, W, G)
            power_loading_new = climb_constraint_power
            P_new = 1/power_loading_new * W
            if abs(P_new - P[i]) <= tolerance:
                converged = True
            P[i] = P_new
    return S, P


def cruise_constraint_converge(x, y, e, AR, v_cruise, eta_p, q_cruise, cd_0):
    S = np.linspace(200, 600, 1000)
    P = np.zeros_like(S)
    for i in range(len(S)):
        S_0 = S[i]
```

```
        P[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S_0, P[i], S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading = W/S_0
            cruise_constraint, cruise_constraint_power = cg.cruise_constraint(e, AR, W, W,
wing_loading, v_cruise, eta_p, q_cruise, cd_0, P[i] * 0.8, P[i])
            power_loading_new = cruise_constraint_power
            P_new = 1/power_loading_new * W
            if abs(P_new - P[i]) <= tolerance:
                converged = True
            P[i] = P_new
    return S, P

def ceiling_constraint_converge(x, y, e, AR, cd_0, rho_cruise, cl_max_cruise, eta_p):
    S = np.linspace(200, 600, 1000)
    P = np.zeros_like(S)
    for i in range(len(S)):
        S_0 = S[i]
        P[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S_0, P[i], S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading = W/S_0
            ceiling_constraint, ceiling_constraint_power = cg.ceiling_constraint(e, AR, cd_0,
rho_cruise, cl_max_cruise, wing_loading, eta_p, W, W, P[i] * 0.8, P[i])
            power_loading_new = ceiling_constraint_power
            P_new = 1/power_loading_new * W
            if abs(P_new - P[i]) <= tolerance:
                converged = True
            P[i] = P_new
    return S, P


def maneuver_constraint_converge(x, y, e, AR, v_stall_flight, R_turn, q_cruise, cd_0,
rho_ground, cl_max, eta_p):
    S = np.linspace(200, 600, 1000)
    P = np.zeros_like(S)
    for i in range(len(S)):
        S_0 = S[i]
        P[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S_0, P[i], S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading = W/S_0
            maneuver_constraint, maneuver_constraint_power, n = cg.maneuver_constraint(e, AR,
v_stall_flight, R_turn, W, W, q_cruise, cd_0, wing_loading, rho_ground, cl_max, eta_p,
P[i]*0.8, P[i])
            power_loading_new = maneuver_constraint_power
            P_new = 1/power_loading_new * W
            if abs(P_new - P[i]) <= tolerance:
                converged = True
```

```
            P[i] = P_new
    return S, P

def stall_constraint_converge(x, y, v_stall, rho_cruise, cl_max):
    P = np.linspace(150, 900, 1000)
    S = np.zeros_like(P)
    for i in range(len(P)):
        P_0 = P[i]
        S[i] = 300
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S[i], P_0, S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading_stall = cg.stall_constraint(v_stall, rho_cruise, cl_max)
            S_new = W / wing_loading_stall
            if abs(S_new - S[i]) <= tolerance:
                converged = True
            S[i] = S_new
    return S, P

def landing_constraint_converge(x, y, s_L, s_a, cl_max_land, rho_ground, rho_sea):
    P = np.linspace(150, 900, 1000)
    S = np.zeros_like(P)
    for i in range(len(P)):
        P_0 = P[i]
        S[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S[i], P_0, S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading_landing = cg.landing_constraint(s_L, s_a, cl_max_land, rho_ground,
rho_sea, W - 4000, W)
            S_new = W / wing_loading_landing
            if abs(S_new - S[i]) <= tolerance:
                converged = True
            S[i] = S_new
    return S, P

def stall_constraint_maneuver_converge(x, y, rho_cruise, v_stall_flight, cl_max_TO, n):
    P = np.linspace(150, 900, 1000)
    S = np.zeros_like(P)
    for i in range(len(P)):
        P_0 = P[i]
        S[i] = 200
        tolerance = .001
        converged = False
        while converged == False:
            results = we.solve_takeoff_weight_2(S[i], P_0, S_design = 12214.12 / x, P_design =
12214.12 / y)
            takeoff_weight, empty_weight_frac, fuel_weight_frac,\
            empty_weight, fuel_weight, iterations = results
            W = takeoff_weight
            wing_loading_landing = cg.stall_constraint_maneuver(rho_cruise, v_stall_flight,
cl_max_TO, n, W, W)
            S_new = W / wing_loading_landing
            if abs(S_new - S[i]) <= tolerance:
                converged = True
            S[i] = S_new
    return S, P
```

```
S_takeoff, P_takeoff, = takeoff_constraint_converge(x, y, s_TO=1000, rho_ground=.002378,
rho_sea=.002378, cl_max_TO=1.7333, eta_p=0.9)
S_climb1, P_climb1 = climb_constraint_converge(x, y, e=1.03, AR=5, k_s=1.3, cd_0=0.0436,
cl_max_climb=1.7333, rho=.002378, eta_p=0.9, G=0.04, type='takeoff')
S_climb2, P_climb2 = climb_constraint_converge(x, y, e=0.97, AR=5, k_s=1.3, cd_0=0.1136,
cl_max_climb=2.0, rho=.002378, eta_p=0.9, G=0.03, type='balked')
S_cruise, P_cruise = cruise_constraint_converge(x, y, e=1.10, AR=5, v_cruise=212, eta_p=0.9,
q_cruise=(1/2*.002048 * 212**2), cd_0=0.02862)
S_ceiling, P_ceiling = ceiling_constraint_converge(x, y, e=1.10, AR=5, cd_0=0.02862,
rho_cruise=.002048, cl_max_cruise=1.6, eta_p=0.9)
S_maneuver, P_maneuver = maneuver_constraint_converge(x, y, e=1.10, AR=5, v_stall_flight=145,
R_turn=1800, q_cruise=(1/2*.002048 * 212**2), cd_0=0.02862, rho_ground=.002378, cl_max=1.6,
eta_p=0.9)
S_stall, P_stall = stall_constraint_converge(x, y, v_stall=145, rho_cruise=0.002048,
cl_max=1.6)
S_landing, P_landing = landing_constraint_converge(x, y, s_L=1500, s_a=500, cl_max_land=2.0,
rho_ground=.002378, rho_sea=.002378)
S_stall_maneuver, P_stall_maneuver = stall_constraint_maneuver_converge(x, y,
rho_cruise=.002048, v_stall_flight=145, cl_max_TO=1.6, n=(np.sqrt((145**2 / 1800 / 32.2)**2 +
1)))


plt.figure()
plt.plot(S_takeoff, P_takeoff/.77, label='Takeoff')
plt.plot(S_climb1, P_climb1/.77, label='Takeoff Climb')
plt.plot(S_climb2, P_climb2/.77, label='Balked Climb')
plt.plot(S_cruise, P_cruise/.77, label='Cruise')
plt.plot(S_ceiling, P_ceiling/.77, label='Ceiling')
plt.plot(S_maneuver, P_maneuver/.77, label='Maneuver')
plt.plot(S_stall, P_stall/.77, label='Stall')
plt.plot(S_landing, P_landing/.77, label='Landing Distance')
plt.plot(S_stall_maneuver, P_stall_maneuver/.77, label='Maneuver Stall')

plt.axis([250, 450, 300, 900])
x1 = 5
y1 = 5
plt.scatter(387.2, 620, color='red', s=70, zorder=2)
plt.text(387.2+x1, 620, 'Ceres-100')
plt.scatter(306, 680, color='red', s=70, zorder=2)
plt.text(306+x1, 680, 'AT-402B')
plt.scatter(365, 750, color='red', s=70, zorder=2)
plt.text(365+x1, 750, 'Thrush 510P2')
plt.scatter(294, 750, color='red', s=70, zorder=2)
plt.text(294+x1, 750, 'PAC Cresco')

plt.legend(loc=(.7, .05))
plt.grid()
plt.title("Shaft Power ($P_{shaft}$) [hp] vs. Wing Area ($S$) [ftÂ˛]")
plt.xlabel("$S$ [ftÂ˛]")
plt.ylabel("$P_{shaft}$ [hp]")
plt.show()
```