```
import itertools
import math
# import statsmodels.api as sm
import numpy as np
import matplotlib.pyplot as plt
import csv
# Outputs: stress and compare it to critical stress

def Euler(c, E, Lp):
    return c * (math.pi**2 * E) / Lp**2

def Johnson(c, E, Lp, y_sig):
    return y_sig * (1 - (y_sig * Lp**2)/(4*c*math.pi**2 * E))

def Gerard(kc, E, v, t, b):
    return ((math.pi**2 * kc * E) / (12*(1-v**2))) * (t/b)**2



# 5 DVs. 3 Levels. n_stiff is discrete
# Bounds for DVs
tskin = [0.05, 0.1, .15]
tstiff = [0.05, 0.1, .15]
nstiff = [5, 10, 15]
hstiff = [0.05, 0.1, 0.15]
wstiff = [0.05, 0.1, 0.15]

# Getting Inertias
# hstiff = 2
# wstiff = 1
# tstiff = .01
# tskin = .02
# nstiff = 3

def getStress(hstiff, wstiff, tstiff, tskin, nstiff):
    # Given Constants
    L = 90

    A1 = wstiff * tstiff
    A2 = abs(tstiff * (hstiff-tstiff))

    cen_1x = wstiff/2
    cen_1y = tstiff/2

    cen_2x = tstiff/2
    cen_2y = (hstiff-tstiff)/2 + tstiff

    print(hstiff, wstiff, tstiff, tskin, nstiff)
    print("Areas:", A1, A2)
    y_tot = (wstiff*tstiff * tstiff/2 + (hstiff-tstiff)*tstiff * ((hstiff-tstiff)/2
```

```
+ tstiff)) / (A1 + A2)
    x_tot = (wstiff*tstiff * wstiff/2 + (hstiff-tstiff)*tstiff * (tstiff/2))/ (A1 +
A2)
    print("y_tot = ", y_tot)
    print("x_tot = ", x_tot)

    xh1 = y_tot - cen_1y
    xh2 = cen_2y - y_tot
    Ixx1 = (wstiff*tstiff**3)/12 + A1*xh1**2
    Ixx2 = (tstiff*(hstiff-tstiff)**3)/12 + A2*xh2**2
    Ixx_tot = Ixx1 + Ixx2

    print("Ixx1 = ", Ixx1)
    print("Ixx2 = ", Ixx2)
    print("Ixx_tot = ", Ixx_tot)

    # yh1 = x_tot - cen_1x
    # yh2 = cen_2x - x_tot
    # Iyy1 = (tstiff*wstiff**3)/12 + A1*yh1**2
    # Iyy2 = ((hstiff-tstiff)*tstiff**3)/12 + A2*yh2**2
    # Iyy_tot = Iyy1 + Iyy2

    # print("Iyy1 = ", Iyy1)
    # print("Iyy2 = ", Iyy2)
    # print("Iyy_tot = ", Iyy_tot)

    # Find Smallest Inertia
    min_I = Ixx_tot

    # Find slenderness ratio
    c = 4
    E = 11 * 10**6
    y_sig = 50 * 10**3
    Lp = math.pi * math.sqrt(2 * c * E/y_sig)
    print("\nSlenderness Ratio at E=J: ", Lp)

    # Solve for radius of gyration
    rho = math.sqrt(min_I/(A1 + A2))
    print("Radius of Gyration: ", rho)

    # Find column slenderness ratio
    Lp_col = L / rho
    print("Column slenderness ratio: ", Lp_col)

    colCr = 0

    # Compare slenderness ratio to column slenderness ratio and find crit buckling
stress
    if(Lp_col > Lp):
        print("\nUsing Euler's Solution to find critical buckling stress")
```

```
    colCr = Euler(c, E, Lp_col)
else:
    print("\nUsing Johnson's Solution to find critical buckling stress")
    colCr = Johnson(c, E, Lp_col, y_sig)

print("Critical buckling stress: ", colCr)

# For the thin plate
Kc = 7.2
v = .3
wDomain = 60 / nstiff

plateCr = Gerard(Kc, E, v, tskin, wDomain)
print("\nUsing Gerard's Solution to find Plate critical buckling stress")
print("Critical Buckling Stress: ", plateCr)

# Solving for Fstiff and Fskin

Astiff = A1 + A2
Askin = wDomain * tskin

Fstiff = 40 * wDomain * Astiff / (Astiff + Askin)
print("\nF_stiff: ", Fstiff)

Fskin = 40 * wDomain * Askin / (Askin + Astiff)
print("F_skin: ", Fskin)

# Turn Critical Stress into Critical Load
# Column Stress to load
colLoad = colCr * Astiff
print("Critical Load of Stiffener: ", colLoad)

plateLoad = plateCr * Askin
print("Critical Load of Plate: ", plateLoad)

colMass = L * Astiff * .1
plateMass = L * Askin * .1
totalMass = colMass * nstiff + plateMass

# Actual Stress
StressStiff = Fstiff / Astiff
StressSkin = Fskin / Askin


minStress = min(StressSkin, StressStiff)

return totalMass, minStress
```

```
# getStress(hstiff, wstiff, tstiff, tskin, nstiff)




#   number = 1
MinMass = 0
# Full Factorial
array = list(itertools.product(tskin,tstiff,nstiff,hstiff,wstiff))
with open('DV_data.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    for i in array:
        writer.writerow(i)

# Group Data by max and mins of each DV
minTskin = []
maxTskin = []
minTstiff = []
maxTstiff = []
minNstiff = []
maxNstiff = []
minHstiff = []
maxHstiff = []

# Write to CSV file
# f = open('massOutput.csv', 'w')
# writer = csv.writer(f)
with open('massOutput.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    # Solve for stress/mass
    print(array)
    output = []
    for data in array:
        t_skin = data[0]
        t_stiff = data[1]
        n_stiff = data[2]
        h_stiff = data[3]
        w_stiff = data[4]

        datapt = getStress(h_stiff, w_stiff, t_stiff, t_skin, n_stiff)
        if(t_skin == tskin[0]): # Min
            minTskin.append(datapt[0])
        else:
            maxTskin.append(datapt[0])
        # Output is totalMass, MaxStress
        writer.writerow(datapt)
        output.append(datapt)
    #print(output)
    print(output)
    for i in output:
```

```
        print(i)



f.close()


# Plotting
# fix, ax = plt.subplots(figsize=(10, 6))
# sm.graphics.(range(5), model.params[1:], ax=ax)
# ax.set_title('Factor Effects Chart')
# ax.set_xlabel('Factors')
# ax.set_ylabel('Effects')
# plt.show()
# f.close()

# f = open("StressOutput.txt", 'r')
# data = np.empty([1, 1])
# for i in range(number-1):

#     np.append(data, float(f.readline()))

# print(data)
```