EAE 135 Project 2: Rocket Engine Test Report

Andrew Matel and Nicholas Rusali

March 3rd, 2025

1 Introduction

There is a rocket being tested against the ground with lift being generated to counteract the weight (refer to Figure 2). There is thrust being produced at the tail-end giving 1250kN of force. The lift being generated is 1400kN and the weight of the rocket is 500kN (summarized in Table 2). The cross section of the rocket is simplified as a laminated cylinder, with the outer layer being carbon-fiber-epoxy (CFRP) and the inner layer being aluminum. For the simplicity of the project, the core is assumed to be empty.

The CFRP layer itself is made up of 8 layers with an orientation of the layup being [0/+45/-45/90/90/-45/+45/0], with each number referring to the degrees of orientation (i.e. +45 is +45° of rotation). The dimensions of the fuselage cross section are showed in Table 1. The material properties for both the CFRP and Al are given in the project description and shown in Table 3 and 4 respectively.

The goal of this project is to find the forces acting upon the rocket in both the x_1 direction (labeled P_1) and the x_2 direction (labeled P_2). These forces will then be used to find the axial displacement, axial strain, and axial stress of the fuselage due the axial load only. This is done using the governing equations of a cantilever beam. The rest of the axial stress will then be calculated using the bending moment rather than from the governing equation of the bending moment (this is to simplify the project).

Some of the assumptions that are made are that this is a 3D Euler-Bernoulli beam under axial loading and bending loading. The laminate layers can assumed to have uniform properties (each layer of the CFRP is uniform, but the CFRP layer as a whole is not). Another assumption that is made is that the stiffness is primarily from the CFRP layer and anything under

(towards the core) is negligible (meaning that the aluminum doesn't add comparable stiffness).

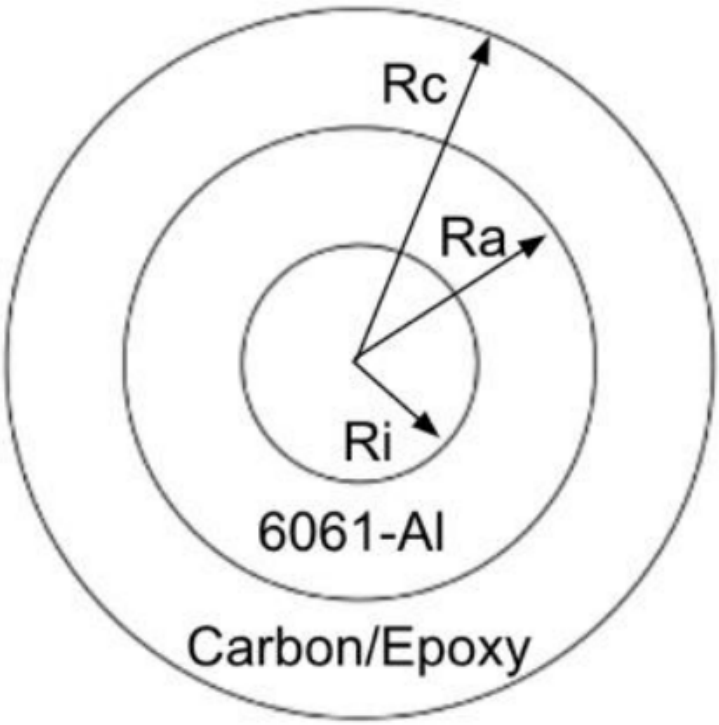

Figure 1: Rocket SRB Fuselage

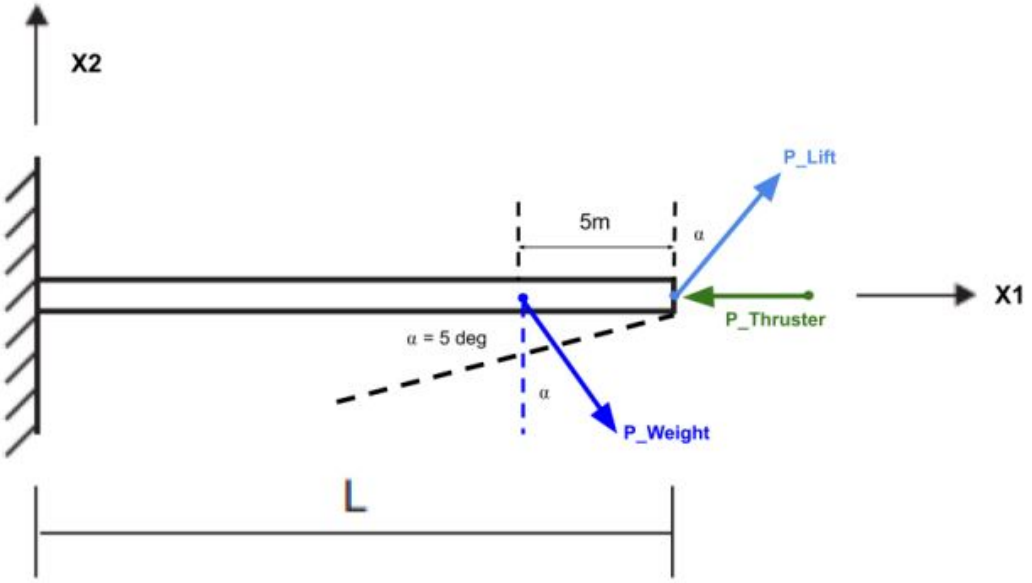

Figure 2: Diagram of the Problem Formulation

Table 1: Fuselage Component Lengths

Component	**Length** $[m]$
Diameter	3
Aluminum/CFRP Thickness	0.005
Fuselage Length	20
R_c	1.500
R_a	1.495
R_i	1.490

Table 2: Forces on Fuselage

Component	**Force** $[kN]$
Thrusters	1250
Lift	1400
Weight	500
P1 (Horizontal Resultant at Tip)	?
P2 (Vertical Resultant at Tip)	?

Table 3: Material Properties of AS4/Epoxy

Property	**Value**
E_{11}	148.24 $[GPa]$
E_{22}	10.07 $[GPa]$
G_{12}	5.58 $[GPa]$
ν_{12}	0.30
Axial Tensile Strength	310 [ksi]
Axial Compressive Strength	-184 [ksi]
Transverse Tensile Strength	7.75 [ksi]
Transverse Compressive Strength	-24.4 [ksi]

Table 4: Material Properties of 6061-T6 Aluminum

Property	**Value**
$(\sigma_s)_T \cong (\sigma_s)_C$	+/- 289.6 $[MPa]$
E_C	10.07 $[GPa]$
E_T	68.95 $[GPa]$
G	26.2 $[GPa]$
ν_{12}	0.30

2 Equations

$$H_{33}^c = \iint Ex_2^2 dA = E_{11}\pi \frac{r_{i+1}^4 - r_i^4}{4} \tag{1}$$

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = [Q] \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{Bmatrix} \tag{2}$$

$$Q = \begin{bmatrix} \frac{E_{11}}{1-\nu_{12}\nu_{21}} & \frac{\nu_{21}E_{11}}{1-\nu_{12}\nu_{21}} & 0 \\ \frac{\nu_{12}E_{11}}{1-\nu_{12}\nu_{21}} & \frac{E_{22}}{1-\nu_{12}\nu_{21}} & 0 \\ 0 & 0 & G_{12} \end{bmatrix} \tag{3}$$

$$\overline{Q} = T^{-1} \cdot Q \cdot R \cdot T \cdot R^{-1} \tag{4}$$

$$T = \begin{bmatrix} \cos^2\beta & \sin^2\beta & 2\sin\beta\cos\beta \\ \sin^2\beta & \cos^2\beta & 2\sin\beta\cos\beta \\ -\sin\beta\cos\beta & \sin\beta\cos\beta & \cos^2\beta - \sin^2\beta \end{bmatrix} \tag{5}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{6}$$

$$\left(\overline{S}\right)_{ith} = \left[\left(\overline{Q}\right)_{ith}\right]^{-1} \tag{7}$$

$$\left(E_{x1x1}\right)_{ith} = \frac{1}{\left(\overline{S}_{11}\right)_{ith}} \tag{8}$$

$$\sigma_1\left(0, x_2, x_3\right) = -\frac{\left(E_{x1x1}\right)_{ith} \cdot x_2 \cdot M_3(0)}{H_{33}^c} \tag{9}$$

3 Results

For our calculations, we split the problem into two parts. The first part deals only with the axial components of the loads and their effect on the axial stress of the structure and the second part only deals with the axial stresses due to bending. The two methods are subtly different.

3.1 Axial

Beginning with the first part, We began with creating the graphs of the internal reactions. To do this, we must know the reactions at the cantilever end of the structure. These are the equations for those reactions and for the resultant forces at the tip of the fuselage:

$$P_{1tip} = P_{Thrust} - P_{Lift}\sin(\alpha) = 144.7kN \tag{10}$$

$$P_{2tip} = P_{Lift}\sin(\alpha) = 122.0kN \tag{11}$$

$$R_1 = P_{Thrust} - P_{Lift}\sin(\alpha) - P_{Weight}\sin(\alpha) = 1084.4kN \tag{12}$$

$$R_2 = -P_{Lift}\cos(\alpha) + P_{Weight}\cos(\alpha) = -896.6kN \tag{13}$$

$$M_3 = P_{Lift}\cos(\alpha)\cdot(L) - P_{Weight}\cos(\alpha)\cdot(L-5) = -20.42E^{+}6Nm \tag{14}$$

With these values, we can plot the internal forces along the structure in the x_1 direction. Because this is a laminate, we have to find the axial stress of each individual layer. There is an open layer, an Aluminum layer, and there are eight carbon epoxy layers. From each layer, we need to find the cross-sectional area and multiply that by the appropriate Young's Modulus to find the axial stiffness of the cross-section. The following equations show this process:

$$A_{CFRP} = \pi\left((R_a + top\cdot t)^2 - (R_a + bot\cdot t)^2\right) \tag{15}$$

$$A_{Al} = \pi\left((R_a)^2 - (R_i)^2\right) \tag{16}$$

Where t is the thickness of each of the composite layers and bot and top are indicators of the bottom and top coordinates of each layers. With these, the following axial stiffness can be derived:

$$S = \Sigma E_i A_i = 5.37GPa \tag{17}$$

Finally, with this stiffness value, we can calculate the stress, strain, and displacement due to the axial components of the external loads. This calculation will use the axial load governing equation:

$$\frac{d}{dx_1}(S\frac{du_1}{dx_1}) = -p_1(x_1) \tag{18}$$

Since our distributed axial load is equal to zero, the result of our integration is:

$$u_1 = \frac{C_1 x_1}{S} + C_2 \tag{19}$$

$$\epsilon_1 = \frac{N_1(x_1)}{S} \tag{20}$$

Where the boundary conditions $u_1 = 0$ and $N_1(x_1 = 0) = R_1$ dictate that C_1 and C_2 are the following values:

$$C_1 = N_{1,x_1=0} = R_1$$

$$C_2 = 0$$

With these values, we can determine the stress along the longitudinal axis of the structure (x_1) using the following equation:

$$\sigma_1 = E\frac{N_1(x_1)}{S} \tag{21}$$

3.2 Bending

Each layer will have a different stress and each layer will have a different Young's modulus, so we will have to compute the Young's modulus for each of the orientations of the composite. To do this, we follow the following procedure:

1. Find the Q matrix (Equation 3)
2. Determine the $\overline{Q}$ of each of the layer orientations using the T matrix (Equation 5), R matrix (Equation 6) and Equation 4
3. Use Equation 7 to compute $\overline{S}$

4. Take the [1,1] value of each $\overline{S}$ and compute E_{11} for each layer using Equation **??**

Using this procedure, we find that:

$$E_{11_{0^\circ}} = 148.24GPa$$
$$E_{11_{-45^\circ}} = 14.22GPa$$
$$E_{11_{45^\circ}} = 14.22GPa$$
$$E_{11_{90^\circ}} = 10.07GPa$$

With these values, we find that the stress at the wall (x_1) follows the trend shown in Figure 7. As we can see from this figure, the outermost layer has the highest stress with a value of **29.95 MPa** in compression. We will be using this value for our failure analysis.

That concludes the axial stress due to axial loading, now we can move onto the axial stress due to bending.

In this section, we use a different method to determine the axial stress. We do not use the governing equations to determine the displacement, we simply use the internal moments, bending stiffness, as well as the previously calculated axial Young's moduli for each of the layers.

The first step in determining the stress is to calculate the bending stiffness of the cross section. To calculate the bending stiffness, we use Equation 1. This calculation yields the following:

$$H^c_{33} = 6.10E^{+}9Nm^2$$

3.3 Results

Now that we have H^c_{33}, we can find the axial stress caused by bending in each of the layers. This uses Equation 9 to find the stresses at the wall with respect to changes in radial distance from the center of the structure. This result is displayed in Figure 7, where we show the compressive stress due to bending. For clarity of the figure, we have only included the outer 0.01 m section. The stress from 0 to 1.49 m is zero since this area is empty. As we can see from the graph, the largest axial stress due to bending occurs at the outermost layer of the tube. This value comes to approximately **743.99 MPa** in compression, which will become important for the failure analysis of the structure.

In the following section, we will cover the procedure used to determine the results found in Figure 10, where we depict the safety factors associated

with each of the layers of the structure at the junction between the structure and the cantilever boundary.

To determine the factors of safety of each of the layers, we will be considering the maximum total axial stress developed in the cross section. Because we are assuming very small deformations, we can use superposition of stress to add up the axial stress due to axial loads and the axial stress due to bending. The result of this addition is shown in Figure. To determine the factor of safety, we take the compressive yield stresses of each of the layers and divide them by the maximum compressive axial stress of the structure. The results of these calculations are displayed in Figure 10. As we can see, when comparing these values to the standards of spacecraft and aircraft structures, all of the layers are determined to be above these values implying that this structure meets safety requirements. The lowest safety factor of all of the layers is approximately **1.64** which corresponds to the CFRP with an orientation of zero degrees. The highest factor of safety is **19.77** which corresponds to the aluminum layer. These results will be discussed further in the discussion section of this report.

4 Graphs

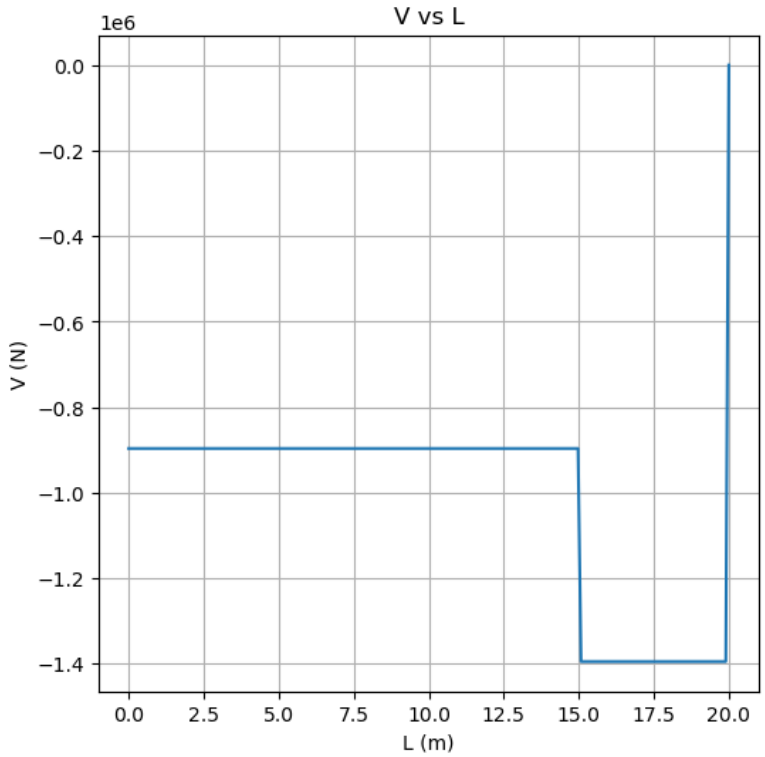

Figure 3: Shear Force Along the Span of the Fuselage

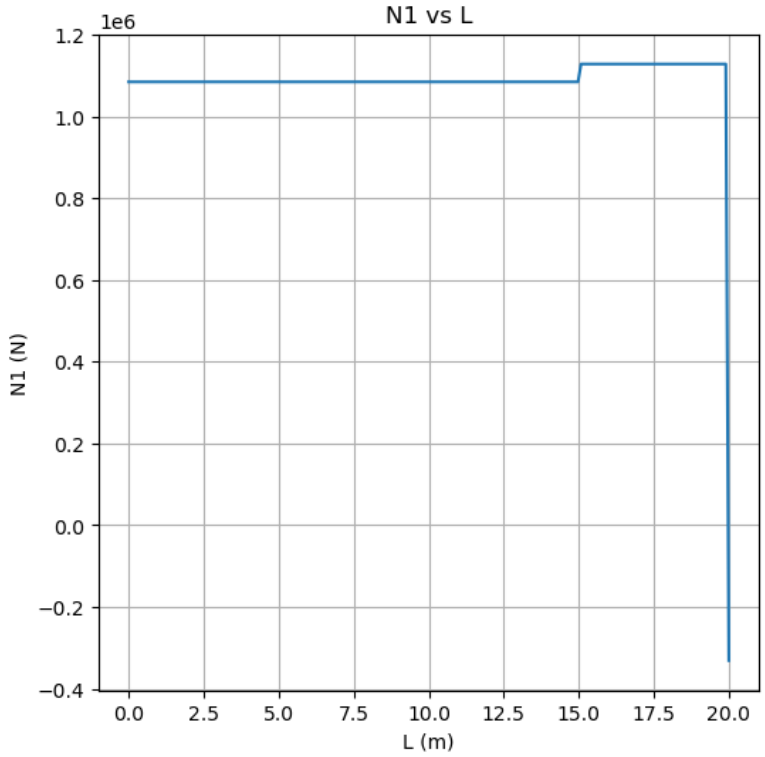

Figure 4: Internal Force in the Axial Direction Along Span

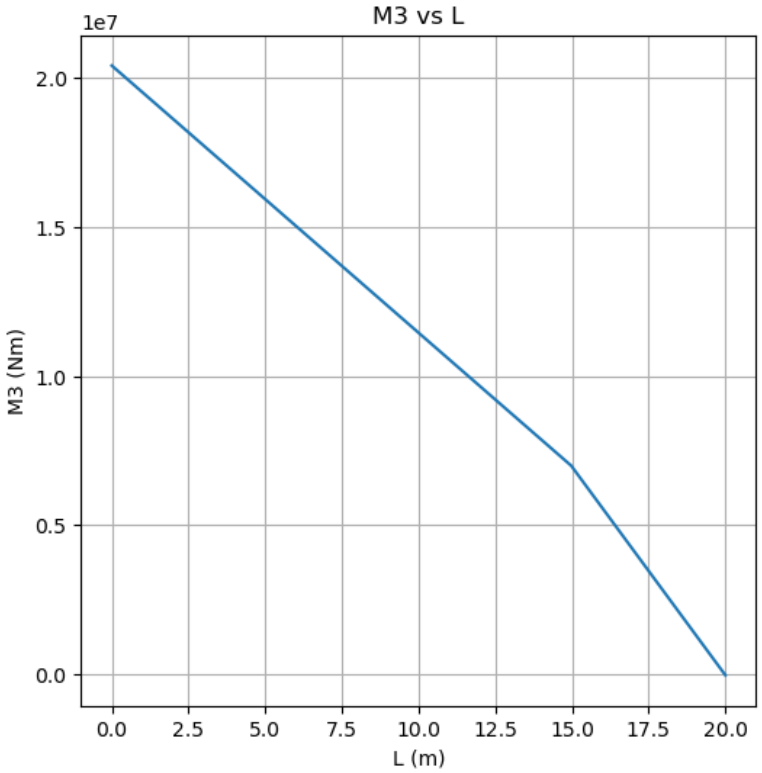

Figure 5: Internal Moment Along the Span of the Fuselage

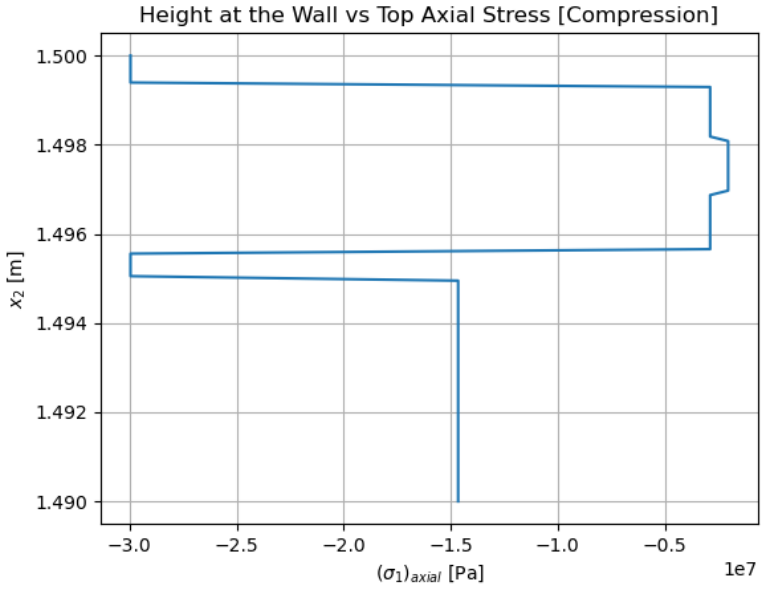

Figure 6: Plot of Top Axial Stress

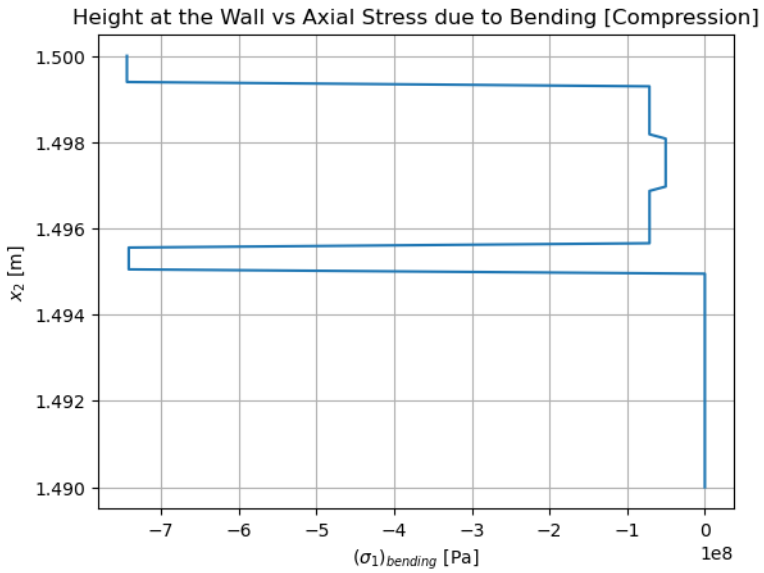

Figure 7: Plot of Axial Stress due to Bending (Compression)

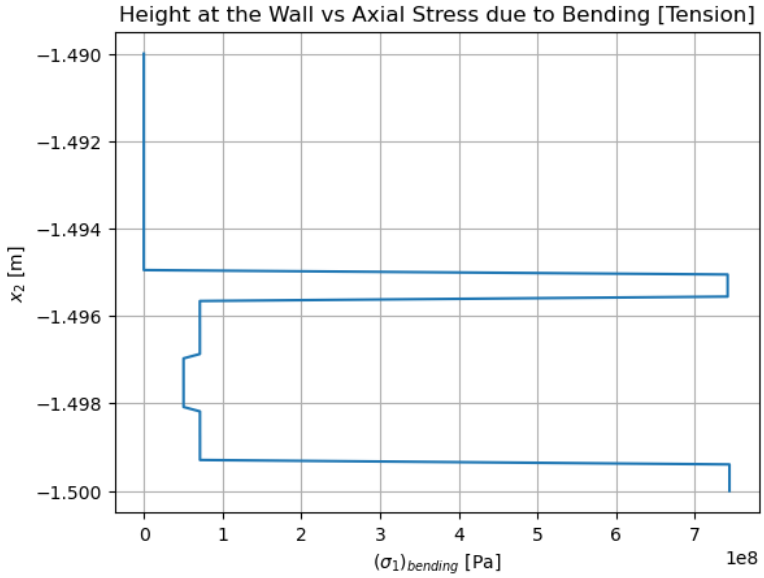

Figure 8: Plot of Axial Stress due to Bending (Tension)

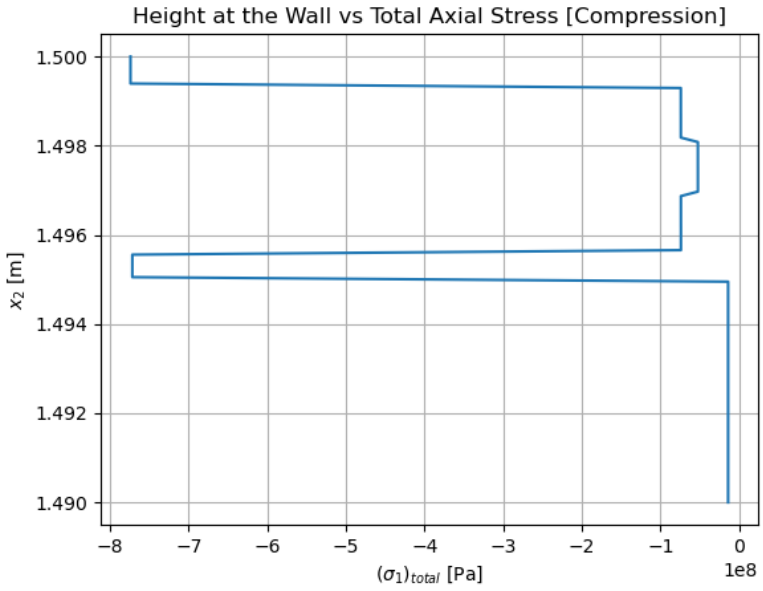

Figure 9: Plot of Total Axial Stress due to Axial Load and Bending

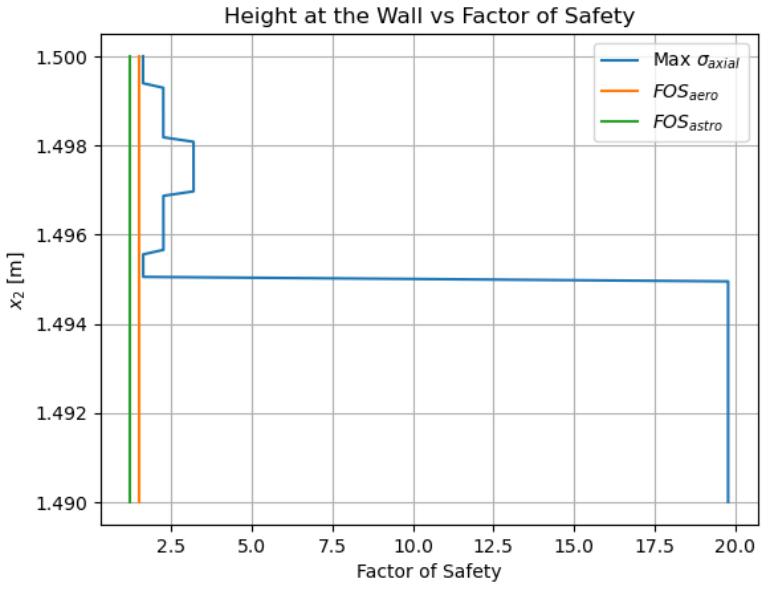

Figure 10: Plot of the Factor of Safety Along the Wall

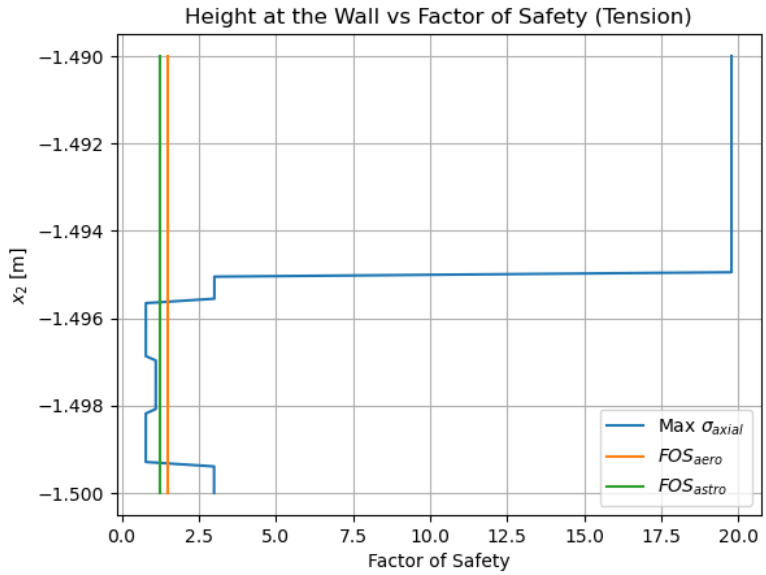

Figure 11: Plot of the Factor of Safety Along the Wall (Tension)

5 Discussion

Looking at these results, we see that the aluminum layer easily passes the safety requirements for both the spacecraft and aircraft applications. On the other hand, the outermost CFRP layer only barely passes the safety requirement for aircraft application under compression, but it is more suitable when considering the spacecraft application.

When analyzing the fuselage under tensile load, we find that the CFRP layer, specifically the +/-45° and 90° orientated layup, do not pass factor of safety for either requirements.

Due to these results, we recommend that the aluminum layer should be made less thick in order to reduce weight and cost while still passing safety requirements and to add more layers in the 0° orientation. While the 0° layer does not contribute much to the factor of safety under compression, it has high tensile strength when compared to the 45° and 90° orientations. Another possible solution is to weave the CFRP rather than using a layup in order to achieve the require compressive and tensile strength.

6 Appendix

6.1 Video

Link to video submission:
https://video.ucdavis.edu/media/EAE135+Project+2+Recording+2+Andrew+Matel++Nicholas+Rusali/1_hv37qxjw

6.2 Hand Calculations

Next page

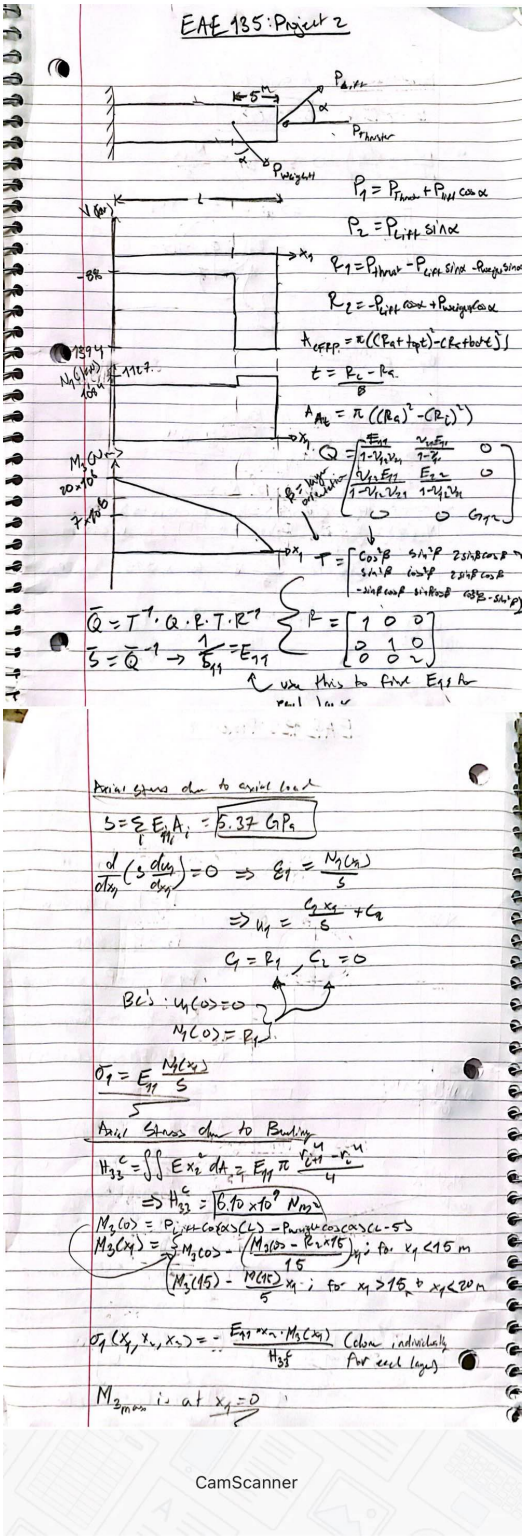

Figure 12: Hand Calculations

6.3 Code

In [33]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

In [34]:
```
#EAE 135 Project 2
#Andrew Matel
#Nicholas Rusali

### Given Info
## Lengths
D = 3 #m
t_layers = 0.005 #m
L = 20 #m
Ri = 1.490 #m
Ra = 1.495 #m
Rc = 1.5 #m
num = 200
x1 = np.linspace(0, L, num)
## Forces
Thrust = 1250*(10**3) #N
Lift = 1400*(10**3) #N
Weight = 500*(10**3) #N
alf = math.radians(5) #rads

## Material Properties
#CFRP
E11_cfrp = 148.24*(10**9) #Pa
E22_cfrp = 10.07*(10**9) #Pa
G12_cfrp = 5.58*(10**9) #Pa
v12_cfrp = 0.30
#Al 6061
syc_Al = -289.6 #MPa (yield stress in compression)
syt_Al = 289.6 #MPa
Ec_Al = 69.69*(10**9) #Pa
Et_Al = 68.95*(10**9) #Pa
G_Al = 26.2*(10**9) #Pa
v_Al = 0.33

### Reactions at Cantilever
R1 = Thrust - (Lift+Weight)*(math.sin(alf))
R2 = (Weight - Lift)*math.cos(alf)
M3 = (Weight*(L-5)-Lift*L)*math.cos(alf)

print('R1 = {} N'.format(R1))
print('R2 = {} N'.format(R2))
print('M3 = {} Nm'.format(M3))
```

Figure 13: Screenshot of Code Part 1

```
### Internal Force Calculations
Shear = np.zeros(num)
for i in range(0,len(Shear)):
    if (x1[i]<15):
        Shear[i] = R2
    elif (x1[i]>=15) & (x1[i]<L):
        Shear[i] = R2 - Weight*(math.cos(alf))
    else:
        Shear[i] = R2 - (Weight - Lift)*math.cos(alf)

plt.figure(figsize = [6,6])
plt.title('V vs L')
plt.xlabel('L (m)')
plt.ylabel('V (N)')
plt.plot(x1, Shear)
plt.grid()

N1 = np.zeros(num)
for i in range(0,len(N1)):
    if (x1[i]<15):
        N1[i] = R1
    elif (x1[i]>=15) & (x1[i]<L):
        N1[i] = R1 + Weight*(math.sin(alf))
    else:
        N1[i] = R1 - Thrust - (Lift+Weight)*(math.sin(alf))

plt.figure(figsize = [6,6])
plt.title('N1 vs L')
plt.xlabel('L (m)')
plt.ylabel('N1 (N)')
plt.plot(x1, N1)
plt.grid()

slope1 = (abs(M3)-abs(R2*15)-abs(M3))/15
slope2 = 6995889.902428213/5
istart = 149
Mom = np.zeros(num)
for i in range(0,len(Mom)):
    if (x1[i]<15):
        Mom[i] = abs(M3) + (slope1)*(x1[i])
    elif (x1[i]>=15):
        Mom[i] = Mom[istart] - (slope2)*((x1[i]-x1[istart]))

print(Mom[istart])
plt.figure(figsize = [6,6])
plt.title('M3 vs L')
plt.xlabel('L (m)')
plt.ylabel('M3 (Nm)')
plt.plot(x1, Mom)
plt.grid()
```

Figure 14: Screenshot of Code Part 2

In [35]:

```
###Axial Problem
x1 = np.linspace(0,20,20)
##Stiffness

#Al
A_Al = (math.pi)*((Ra**2)-(Ri**2))

#CFRP
#There is a different area occupied by each of the 8 layers:
t_cfrp_layers = (Rc-Ra)/8 #radial thickness of layers
A_0_1 = (math.pi)*(-(Ra**2)+(Ra+t_cfrp_layers)**2)
A_45_1 = (math.pi)*(-((Ra+t_cfrp_layers)**2)+(Ra+2*t_cfrp_layers)**2)
A_neg45_1 = (math.pi)*(-((Ra+2*t_cfrp_layers)**2)+(Ra+3*t_cfrp_layers)**2)
A_90_1 = (math.pi)*(-((Ra+3*t_cfrp_layers)**2)+(Ra+4*t_cfrp_layers)**2)
A_90_2 = (math.pi)*(-((Ra+4*t_cfrp_layers)**2)+(Ra+5*t_cfrp_layers)**2)
A_neg45_2 = (math.pi)*(-((Ra+5*t_cfrp_layers)**2)+(Ra+6*t_cfrp_layers)**2)
A_45_2 = (math.pi)*(-((Ra+6*t_cfrp_layers)**2)+(Ra+7*t_cfrp_layers)**2)
A_0_2 = (math.pi)*(-((Ra+7*t_cfrp_layers)**2)+(Ra+8*t_cfrp_layers)**2)

#axial stiffness
S = Ec_Al*A_Al + E11_cfrp*(A_0_1 + A_0_2) +E22_cfrp*(A_45_1+A_45_2+A_neg45_1+A_neg45_2+A_90_1+A_90_2)
print(S)

##Displacement
#Displacement in axial direction due to the axial forces (using govering eqn)
#(d/dx1[S*(du1/dx1)]=-p1=0)

#Boundary Conditions
C1 = R1 #When x1 = 0, N1 = R1
C2 = 0 #no displacement at root
u1 = (C1*x1)/S + C2

##Strain
#Axial strain due to axial load only N1 = axialstrain*S
axial_strain1 = np.zeros(len(N1))
for i in range(0, len(axial_strain1)):
    axial_strain1[i] = (N1[i])/S

##Stress
#Axial Stress due to axial load only
#stress = E*(N1)/S
axial_stress_Al = (Ec_Al*N1)/S
print(axial_stress_Al)
#--> Continued after Bending Section (needed to find Ex1x1 in that section)
```

Figure 15: Screenshot of Code Part 3

In [36]:

```
#Bending Problem
v21_cfrp = E22_cfrp*(v12_cfrp/(E11_cfrp)) #relationship between v21 and v12

Q = np.array([[(E11_cfrp)/(1-v12_cfrp*v21_cfrp), (v21_cfrp*E11_cfrp)/(1-v12_cfrp*v21_cfrp), 0],
     [(v12_cfrp*E22_cfrp)/(1-v12_cfrp*v21_cfrp), E22_cfrp/(1-v12_cfrp*v21_cfrp), 0],
     [0,0,G12_cfrp]])

def makeTmatrix(beta):
        #This function takes each of the angles of the CFRP and produces a rotation matrix T
        T = np.array([[(math.cos(beta))**2,(math.sin(beta))**2, 2*math.sin(beta)*math.cos(beta)],
                      [(math.sin(beta))**2, (math.cos(beta))**2, -2*math.sin(beta)*math.cos(beta)],
                      [-math.sin(beta)*math.cos(beta), math.sin(beta)*math.cos(beta), (math.cos(beta))**2 - (math.
        return T

T_0 = makeTmatrix(0)
T_neg45 = makeTmatrix(-((math.pi)/4))
T_45 = makeTmatrix((math.pi)/4)
T_90 = makeTmatrix(((math.pi)/2))

R = np.array([[1,0,0],[0,1,0],[0,0,2]]) #Reuter's Matrix
def makeQbar(T, R, Q):
        #This function transforms Q into Qbar using the T and R matrices
        Qbar = np.matmul(np.linalg.inv(T), Q)
        Qbar = np.matmul(Qbar,R)
        Qbar = np.matmul(Qbar,T)
        Qbar = np.matmul(Qbar,np.linalg.inv(R))
        return Qbar

Qbar_0 = makeQbar(T_0, R, Q)
Qbar_neg45 = makeQbar(T_neg45, R, Q)
Qbar_45 = makeQbar(T_45, R, Q)
Qbar_90 = makeQbar(T_90, R, Q)

#Here, we define the Sbar matrix which is the inverse of the Qbar matrix
Sbar_0 = np.linalg.inv(Qbar_0)
Sbar_neg45 = np.linalg.inv(Qbar_neg45)
Sbar_45 = np.linalg.inv(Qbar_45)
Sbar_90 = np.linalg.inv(Qbar_90)

#All we have to do to solve for Ex1x1 is to take the (1,1) term of each Sbar matrix and divde 1 by it
Ex1x1_0 = 1/(Sbar_0[0,0])
Ex1x1_neg45 = 1/(Sbar_neg45[0,0])
Ex1x1_45 = 1/(Sbar_45[0,0])
Ex1x1_90 = 1/(Sbar_90[0,0])


#H33c = w*sum(Ex1x1*(x2i+1**3-x2i**3)/3)
#To allow for this equation to function for circular cross-sections, I have taken the average circumference of eac
#The Modulus-Weighted Centroid is at the center of the tube

Ex1x1_cfrp = [Ex1x1_0, Ex1x1_45, Ex1x1_neg45, Ex1x1_90, Ex1x1_90, Ex1x1_neg45, Ex1x1_45, Ex1x1_0]
```

Figure 16: Screenshot of Code Part 4

```
layer_H33c = np.zeros(8)
for i in range(0, len(layer_H33c)):
        #Layer_H33c[i] = (math.pi)*((Ri + i*t_cfrp_layers + Ri + (i+1)*t_cfrp_layers)/2)*(Ex1x1_cfrp[i])*((((Ri+(i
        layer_H33c[i] = Ex1x1_cfrp[i]*(math.pi)*(((Ra+(i+1)*t_cfrp_layers)**4)-((Ra+(i)*t_cfrp_layers)**4))/4

H33c_Al = (math.pi)*Ec_Al*(((Ra)**4)-((Ri)**4))/4
H33c = sum(layer_H33c) + H33c_Al
print(H33c)

print(Ex1x1_cfrp)
```

Figure 17: Screenshot of Code Part 5

In [37]:
```
#Finishing Axial Stress due to axial load only
axial_stress_cfrp = np.zeros((8,20))
for i in range(0, len(axial_stress_cfrp)):
    for j in range(0, L):
        axial_stress_cfrp[i,j] = (Ex1x1_cfrp[i]*N1[j])/S
```

In [38]:
```
# Axial Stress due to bending
# sigma1(0,x2,x3) = -((Ex1x2)_ith * x2 * M3(0))/H33c
sigma1_total = []
y_span = np.linspace(Ri,Rc,100)

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        sigma1_total.append(-axial_stress_Al[0])
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        sigma1_total.append(-axial_stress_cfrp[0,0] - (Ex1x1_cfrp[0] * (Ra + 0*t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        sigma1_total.append(-axial_stress_cfrp[1,0] - (Ex1x1_cfrp[1] * (Ra + 1*t_cfrp_layers + t_cfrp_layers/2) *
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        sigma1_total.append(-axial_stress_cfrp[2,0] - (Ex1x1_cfrp[2] * (Ra + 2*t_cfrp_layers + t_cfrp_layers/2) *
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        sigma1_total.append(-axial_stress_cfrp[3,0] - (Ex1x1_cfrp[3] * (Ra + 3*t_cfrp_layers + t_cfrp_layers/2) *
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        sigma1_total.append(-axial_stress_cfrp[4,0] - (Ex1x1_cfrp[4] * (Ra + 4*t_cfrp_layers + t_cfrp_layers/2) *
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        sigma1_total.append(-axial_stress_cfrp[5,0] - (Ex1x1_cfrp[5] * (Ra + 5*t_cfrp_layers + t_cfrp_layers/2) *
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        sigma1_total.append(-axial_stress_cfrp[6,0] - (Ex1x1_cfrp[6] * (Ra + 6*t_cfrp_layers + t_cfrp_layers/2) *
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        sigma1_total.append(-axial_stress_cfrp[7,0] - (Ex1x1_cfrp[7] * (Ra + 7*t_cfrp_layers + t_cfrp_layers/2) *
```

Figure 18: Screenshot of Code Part 6

```python
# due to axial load only
sigma1_axial = []

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        sigma1_axial.append(-max(axial_stress_Al))
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        sigma1_axial.append(-max(axial_stress_cfrp[0]))
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        sigma1_axial.append(-max(axial_stress_cfrp[1]))
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        sigma1_axial.append(-max(axial_stress_cfrp[2]))
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        sigma1_axial.append(-max(axial_stress_cfrp[3]))
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        sigma1_axial.append(-max(axial_stress_cfrp[4]))
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        sigma1_axial.append(-max(axial_stress_cfrp[5]))
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        sigma1_axial.append(-max(axial_stress_cfrp[6]))
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        sigma1_axial.append(-max(axial_stress_cfrp[7]))
```

Figure 19: Screenshot of Code Part 7

```python
# due to bending load only
sigma1_bend = []

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        sigma1_bend.append(0)
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        sigma1_bend.append(- (Ex1x1_cfrp[0] * (Ra + 0*t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        sigma1_bend.append(- (Ex1x1_cfrp[1] * (Ra + 1*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        sigma1_bend.append(- (Ex1x1_cfrp[2] * (Ra + 2*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        sigma1_bend.append(- (Ex1x1_cfrp[3] * (Ra + 3*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        sigma1_bend.append(- (Ex1x1_cfrp[4] * (Ra + 4*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        sigma1_bend.append(- (Ex1x1_cfrp[5] * (Ra + 5*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        sigma1_bend.append(- (Ex1x1_cfrp[6] * (Ra + 6*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        sigma1_bend.append(- (Ex1x1_cfrp[7] * (Ra + 7*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
```

Figure 20: Screenshot of Code Part 8

```
# due to bending in tension
sigma1_ten = []

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        sigma1_ten.append(0)
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        sigma1_ten.append( (Ex1x1_cfrp[0] * (Ra + 0*t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        sigma1_ten.append( (Ex1x1_cfrp[1] * (Ra + 1*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        sigma1_ten.append( (Ex1x1_cfrp[2] * (Ra + 2*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        sigma1_ten.append( (Ex1x1_cfrp[3] * (Ra + 3*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        sigma1_ten.append( (Ex1x1_cfrp[4] * (Ra + 4*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        sigma1_ten.append( (Ex1x1_cfrp[5] * (Ra + 5*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        sigma1_ten.append( (Ex1x1_cfrp[6] * (Ra + 6*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        sigma1_ten.append( (Ex1x1_cfrp[7] * (Ra + 7*t_cfrp_layers + t_cfrp_layers/2) * Mom[0])/H33c)
```

Figure 21: Screenshot of Code Part 9

```
sigma1_ten_max = []

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        sigma1_ten_max.append(-axial_stress_Al[0])
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        sigma1_ten_max.append(-axial_stress_cfrp[0,0] + (Ex1x1_cfrp[0] * (Ra + 0*t_cfrp_layers/2) * Mom[0])/H33c)
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        sigma1_ten_max.append(-axial_stress_cfrp[1,0] + (Ex1x1_cfrp[1] * (Ra + 1*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        sigma1_ten_max.append(-axial_stress_cfrp[2,0] + (Ex1x1_cfrp[2] * (Ra + 2*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        sigma1_ten_max.append(-axial_stress_cfrp[3,0] + (Ex1x1_cfrp[3] * (Ra + 3*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        sigma1_ten_max.append(-axial_stress_cfrp[4,0] + (Ex1x1_cfrp[4] * (Ra + 4*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        sigma1_ten_max.append(-axial_stress_cfrp[5,0] + (Ex1x1_cfrp[5] * (Ra + 5*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        sigma1_ten_max.append(-axial_stress_cfrp[6,0] + (Ex1x1_cfrp[6] * (Ra + 6*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        sigma1_ten_max.append(-axial_stress_cfrp[7,0] + (Ex1x1_cfrp[7] * (Ra + 7*t_cfrp_layers + t_cfrp_layers/2)
```

Figure 22: Screenshot of Code Part 10

In [39]:

```
# Plot of the total axial stress at the wall vs X2

plt.figure()
plt.title('Height at the Wall vs Top Axial Stress [Compression]')
plt.xlabel('$(\\sigma_1)_{axial}$ [Pa]')
plt.ylabel('$x_2$ [m]')
plt.plot(sigma1_axial, y_span)
plt.grid()
plt.show()

plt.figure()
plt.title('Height at the Wall vs Axial Stress due to Bending [Compression]')
plt.xlabel('$(\\sigma_1)_{bending}$ [Pa]')
plt.ylabel('$x_2$ [m]')
plt.plot(sigma1_bend, y_span)
plt.grid()
plt.show()

plt.figure()
plt.title('Height at the Wall vs Axial Stress due to Bending [Tension]')
plt.xlabel('$(\\sigma_1)_{bending}$ [Pa]')
plt.ylabel('$x_2$ [m]')
plt.plot(sigma1_ten, -y_span)
plt.grid()
plt.show()

plt.figure()
plt.title('Height at the Wall vs Total Axial Stress [Compression]')
plt.xlabel('$(\\sigma_1)_{total}$ [Pa]')
plt.ylabel('$x_2$ [m]')
plt.plot(sigma1_total, y_span)
plt.grid()
plt.show()

plt.figure()
plt.title('Height at the Wall vs Total Axial Stress [Tension]')
plt.xlabel('$(\\sigma_1)_{total}$ [Pa]')
plt.ylabel('$x_2$ [m]')
plt.plot(sigma1_ten_max, -y_span)
plt.grid()
plt.show()
```

Figure 23: Screenshot of Code Part 11

```
In [ ]: # Calculate the factor of safety along the wall position
ksi2pa = 6.895e6 # ksi to pa conversion
XT = 310*ksi2pa # axial tensile strength [Pa]
XC = -184*ksi2pa # axial compressive strength [Pa]
YT = 7.75*ksi2pa # transverse tensile strength [Pa]
YC = -24.4*ksi2pa # transverse compressive strength [Pa]
sigmaS_al = 289.6e6 #Pa

# calculate the FoS based on the max axial stress
# assume YT and YC for both 90 and 45 deg layer configurations
FoS_max = []

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        FoS_max.append(sigmaS_al/max(axial_stress_Al))
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        FoS_max.append(XC/(-max(axial_stress_cfrp[0]) - (Ex1x1_cfrp[0] * (Ra + 0*t_cfrp_layers/2) * Mom[0])/H33c))
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        FoS_max.append(YC/(-max(axial_stress_cfrp[1]) - (Ex1x1_cfrp[1] * (Ra + 1*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        FoS_max.append(YC/(-max(axial_stress_cfrp[2]) - (Ex1x1_cfrp[2] * (Ra + 2*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        FoS_max.append(YC/(-max(axial_stress_cfrp[3]) - (Ex1x1_cfrp[3] * (Ra + 3*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        FoS_max.append(YC/(-max(axial_stress_cfrp[4]) - (Ex1x1_cfrp[4] * (Ra + 4*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        FoS_max.append(YC/(-max(axial_stress_cfrp[5]) - (Ex1x1_cfrp[5] * (Ra + 5*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        FoS_max.append(YC/(-max(axial_stress_cfrp[6]) - (Ex1x1_cfrp[6] * (Ra + 6*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        FoS_max.append(XC/(-max(axial_stress_cfrp[7]) - (Ex1x1_cfrp[7] * (Ra + 7*t_cfrp_layers + t_cfrp_layers/2)
```

Figure 24: Screenshot of Code Part 12

```
FoS_ten = []

for x2 in y_span:
    if x2 <= Ra:
        # calculate axial stress of aluminum due to axial load
        # assume core bending is negligable
        FoS_ten.append(sigmaS_al/max(axial_stress_Al))
    # calculate axial stress of CFRP in bending and add axial stress due to axial load
    elif x2 > Ra and x2 < Ra + t_cfrp_layers:
        # 0 deg
        FoS_ten.append(XT/(-max(axial_stress_cfrp[0]) + (Ex1x1_cfrp[0] * (Ra + 0*t_cfrp_layers/2) * Mom[0])/H33c))
    elif x2 > Ra + t_cfrp_layers and x2 < Ra + 2*t_cfrp_layers:
        # +45 deg
        FoS_ten.append(YT/(-max(axial_stress_cfrp[1]) + (Ex1x1_cfrp[1] * (Ra + 1*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 2*t_cfrp_layers and x2 < Ra + 3*t_cfrp_layers:
        # -45 deg
        FoS_ten.append(YT/(-max(axial_stress_cfrp[2]) + (Ex1x1_cfrp[2] * (Ra + 2*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 3*t_cfrp_layers and x2 < Ra + 4*t_cfrp_layers:
        # 90 deg
        FoS_ten.append(YT/(-max(axial_stress_cfrp[3]) + (Ex1x1_cfrp[3] * (Ra + 3*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 4*t_cfrp_layers and x2 < Ra + 5*t_cfrp_layers:
        # 90 deg
        FoS_ten.append(YT/(-max(axial_stress_cfrp[4]) + (Ex1x1_cfrp[4] * (Ra + 4*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 5*t_cfrp_layers and x2 < Ra + 6*t_cfrp_layers:
        # -45 deg
        FoS_ten.append(YT/(-max(axial_stress_cfrp[5]) + (Ex1x1_cfrp[5] * (Ra + 5*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 6*t_cfrp_layers and x2 < Ra + 7*t_cfrp_layers:
        # +45 deg
        FoS_ten.append(YT/(-max(axial_stress_cfrp[6]) + (Ex1x1_cfrp[6] * (Ra + 6*t_cfrp_layers + t_cfrp_layers/2)
    elif x2 > Ra + 7*t_cfrp_layers and x2:
        # 0 deg
        FoS_ten.append(XT/(-max(axial_stress_cfrp[7]) + (Ex1x1_cfrp[7] * (Ra + 7*t_cfrp_layers + t_cfrp_layers/2)

AeroFOS = np.ones(100)*1.5
AstroFOS = np.ones(100)*1.25
```

Figure 25: Screenshot of Code Part 13

```
In [41]: # plot the FoS along the height
plt.figure()
plt.title('Height at the Wall vs Factor of Safety')
plt.xlabel('Factor of Safety')
plt.ylabel('$x_2$ [m]')
plt.plot(FoS_max, y_span,label='Max $\\sigma_{axial}$')
plt.plot(AeroFOS, y_span,label='$FOS_{aero}$')
plt.plot(AstroFOS, y_span, label='$FOS_{astro}$')
plt.legend()
plt.grid()

# plot the FoS along the height
plt.figure()
plt.title('Height at the Wall vs Factor of Safety (Tension)')
plt.xlabel('Factor of Safety')
plt.ylabel('$x_2$ [m]')
plt.plot(FoS_ten, -y_span,label='Max $\\sigma_{axial}$')
plt.plot(AeroFOS, -y_span,label='$FOS_{aero}$')
plt.plot(AstroFOS, -y_span, label='$FOS_{astro}$')
plt.legend()
plt.grid()
plt.show()
```

Figure 26: Screenshot of Code Part 14