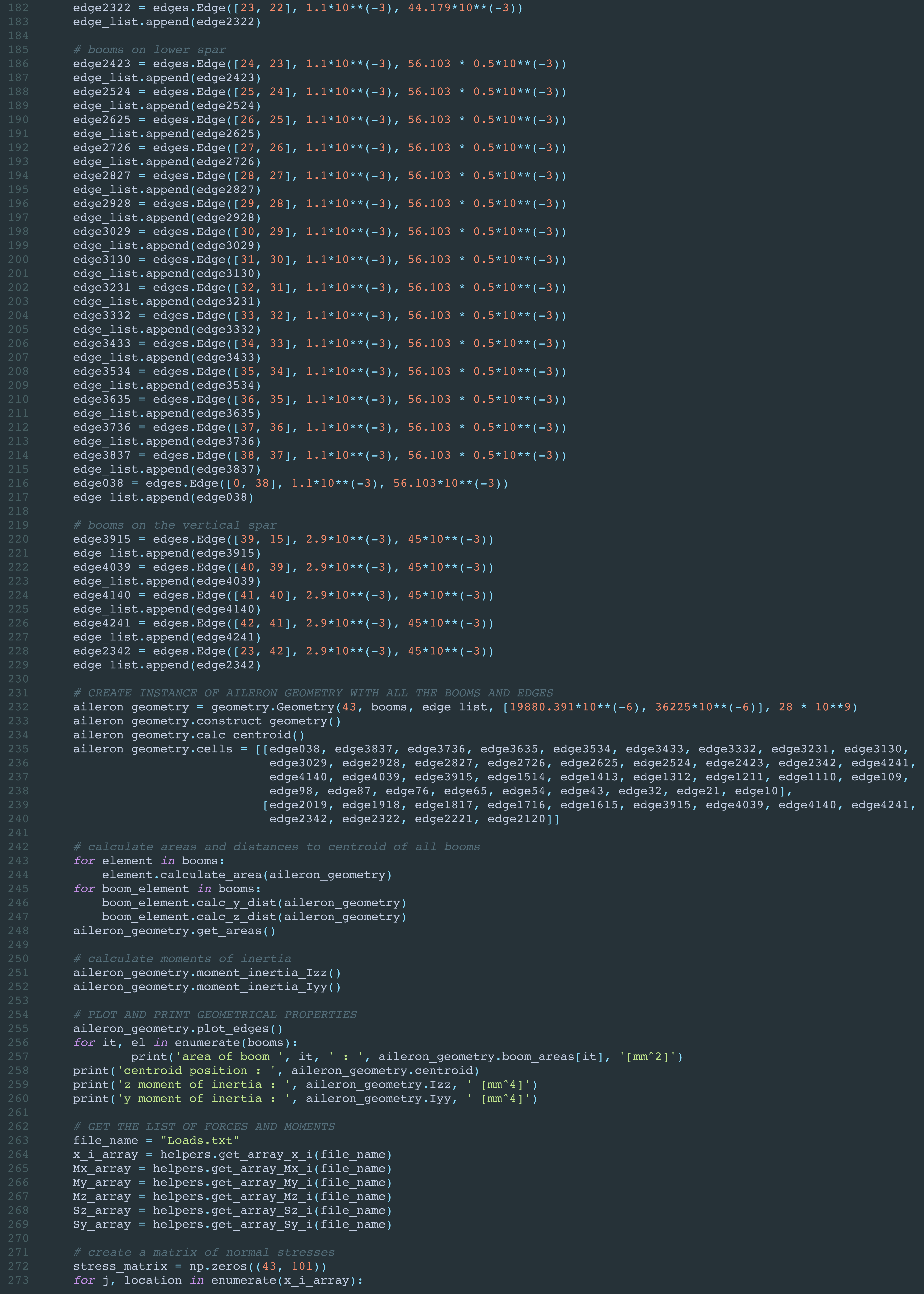
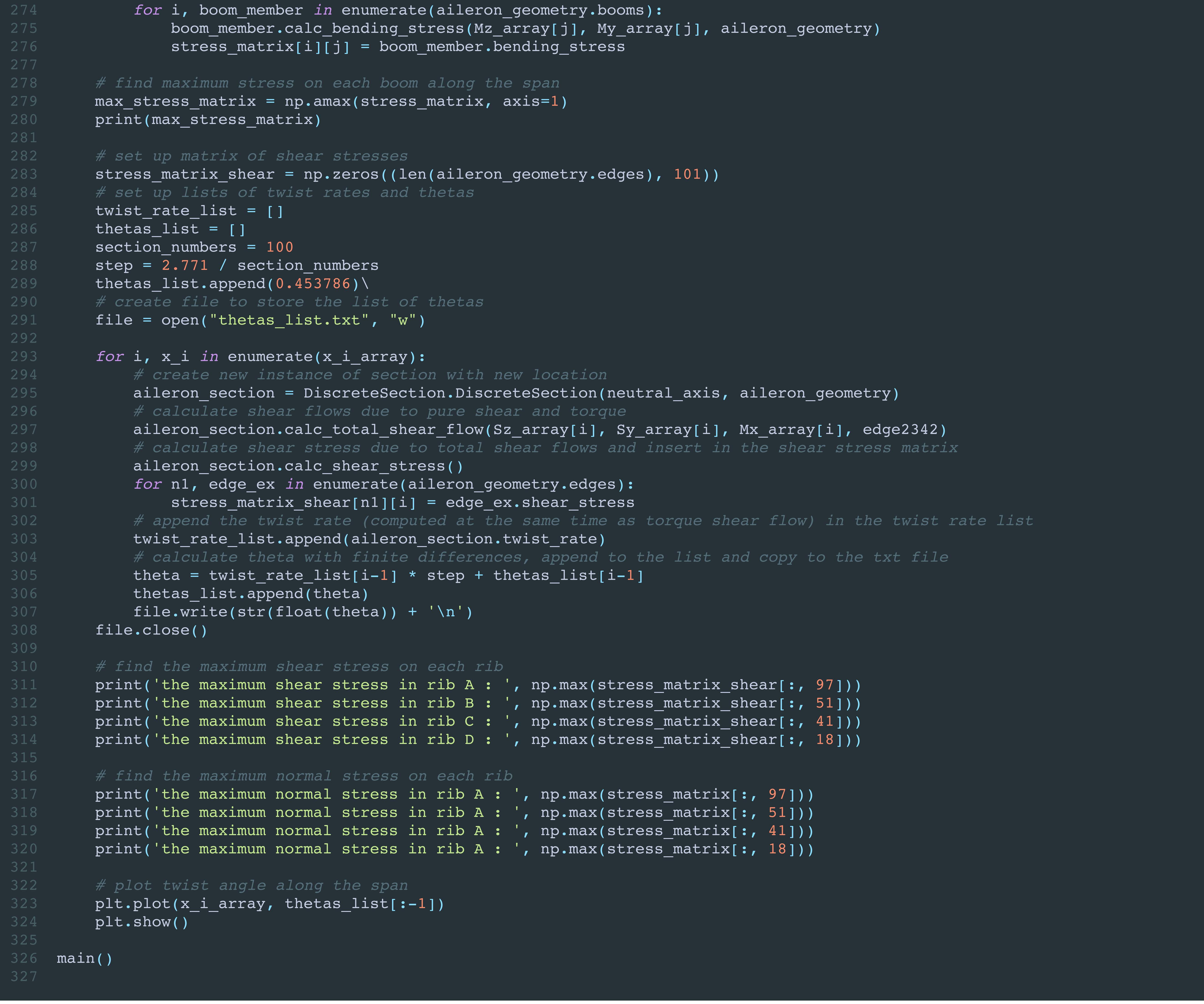
main.py

```python
import unittest
import geometry
import helpers
import numpy as np
import math
import boom
import edges
import DiscreteSection
import matplotlib.pyplot as plt

def main():
    stringer_area = 42 *10**(-6)
    neutral_axis = (0, 1, 0)

    # CREATE LIST OF COORDINATES FOR BOOMS
    # give the coordinates of the booms with respect to the hinge point (origin of coordinates)
    # for the first eight, they are on a straight line
    coordinates = []
    for n in range(16):
        coordinates.append([((43.45 - 5.43125/2 * (n + 1)) * 10) *10**(-3), ((1.40625/2 * (n + 1)) * 10)*10**(-3)])
    # booms 8, 9 and 10 are along a semi-circle
    coordinates.append([-112.5*10**(-3) * math.sin(math.pi / 8), 112.5*10**(-3) * math.cos(math.pi / 8)])
    coordinates.append([-112.5*10**(-3) * math.sin(math.pi / 4), 112.5*10**(-3) * math.cos(math.pi / 4)])
    coordinates.append([-112.5*10**(-3) * math.sin(3 * math.pi / 8), 112.5*10**(-3) * math.cos(3 * math.pi / 8)])
    coordinates.append([-112.5*10**(-3), 0.0])

    # the last 8 are symmetric to the first eight wrt the z-axis
    for i in range(18, -1, -1):
        coords = coordinates[i]
        coordinates.append([coords[0], -coords[1]])
    # the ones on the spar are always at z=0 and distributed equally along the height of the spar
    coordinates.append([0.0, (22.5 + 45)*10**(-3)])
    coordinates.append([0.0, 22.5*10**(-3)])
    coordinates.append([0.0, -22.5*10**(-3)])
    coordinates.append([0.0, (-22.5 - 45)*10**(-3)])


    # CREATE BOOM INSTANCES AND INSERT THEM IN LIST OF BOOMS
    booms = []
    boom0 = boom.Boom(0, coordinates[0], 0.0, neutral_axis)
    booms.append(boom0)
    boom1 = boom.Boom(1, coordinates[1], stringer_area, neutral_axis)
    booms.append(boom1)
    boom2 = boom.Boom(2, coordinates[2], 0.0, neutral_axis)
    booms.append(boom2)
    boom3 = boom.Boom(3, coordinates[3], stringer_area, neutral_axis)
    booms.append(boom3)
    boom4 = boom.Boom(4, coordinates[4], 0.0, neutral_axis)
    booms.append(boom4)
    boom5 = boom.Boom(5, coordinates[5], stringer_area, neutral_axis)
    booms.append(boom5)
    boom6 = boom.Boom(6, coordinates[6], 0.0, neutral_axis)
    booms.append(boom6)
    boom7 = boom.Boom(7, coordinates[7], stringer_area, neutral_axis)
    booms.append(boom7)
    boom8 = boom.Boom(8, coordinates[8], 0.0, neutral_axis)
    booms.append(boom8)
    boom9 = boom.Boom(9, coordinates[9], stringer_area, neutral_axis)
    booms.append(boom9)
    boom10 = boom.Boom(10, coordinates[10], 0.0, neutral_axis)
    booms.append(boom10)
    boom11 = boom.Boom(11, coordinates[11], stringer_area, neutral_axis)
    booms.append(boom11)
    boom12 = boom.Boom(12, coordinates[12], 0.0, neutral_axis)
    booms.append(boom12)
    boom13 = boom.Boom(13, coordinates[13], stringer_area, neutral_axis)
    booms.append(boom13)
    boom14 = boom.Boom(14, coordinates[14], 0.0, neutral_axis)
    booms.append(boom14)
    boom15 = boom.Boom(15, coordinates[15], 0.0, neutral_axis)
    booms.append(boom15)

    # semi circle booms
    boom16 = boom.Boom(16, coordinates[16], 0.0, neutral_axis)
    booms.append(boom16)
    boom17 = boom.Boom(17, coordinates[17], stringer_area, neutral_axis)
    booms.append(boom17)
    boom18 = boom.Boom(18, coordinates[18], 0.0, neutral_axis)
    booms.append(boom18)
    boom19 = boom.Boom(19, coordinates[19], stringer_area, neutral_axis)
    booms.append(boom19)
    boom20 = boom.Boom(20, coordinates[20], 0.0, neutral_axis)
    booms.append(boom20)
    boom21 = boom.Boom(21, coordinates[21], stringer_area, neutral_axis)
    booms.append(boom21)
    boom22 = boom.Boom(22, coordinates[22], 0.0, neutral_axis)
    booms.append(boom22)

    # lower straight line booms
    boom23 = boom.Boom(23, coordinates[23], 0.0, neutral_axis)
    booms.append(boom23)
    boom24 = boom.Boom(24, coordinates[24], 0.0, neutral_axis)
    booms.append(boom24)
    boom25 = boom.Boom(25, coordinates[25], stringer_area, neutral_axis)
    booms.append(boom25)
    boom26 = boom.Boom(26, coordinates[26], 0.0, neutral_axis)
    booms.append(boom26)
    boom27 = boom.Boom(27, coordinates[27], stringer_area, neutral_axis)
    booms.append(boom27)
    boom28 = boom.Boom(28, coordinates[28], 0.0, neutral_axis)
    booms.append(boom28)
    boom29 = boom.Boom(29, coordinates[29], stringer_area, neutral_axis)
    booms.append(boom29)
    boom30 = boom.Boom(30, coordinates[30], 0.0, neutral_axis)
    booms.append(boom30)
    boom31 = boom.Boom(31, coordinates[31], stringer_area, neutral_axis)
    booms.append(boom31)
    boom32 = boom.Boom(32, coordinates[32], 0.0, neutral_axis)
    booms.append(boom32)
    boom33 = boom.Boom(33, coordinates[33], stringer_area, neutral_axis)
    booms.append(boom33)
    boom34 = boom.Boom(34, coordinates[34], 0.0, neutral_axis)
    booms.append(boom34)
    boom35 = boom.Boom(35, coordinates[35], stringer_area, neutral_axis)
    booms.append(boom35)
    boom36 = boom.Boom(36, coordinates[36], 0.0, neutral_axis)
    booms.append(boom36)
    boom37 = boom.Boom(37, coordinates[37], stringer_area, neutral_axis)
    booms.append(boom37)
    boom38 = boom.Boom(38, coordinates[38], 0.0, neutral_axis)
    booms.append(boom38)

    # booms on the spar
    boom39 = boom.Boom(39, coordinates[39], 0.0, neutral_axis)
    booms.append(boom39)
    boom40 = boom.Boom(40, coordinates[40], 0.0, neutral_axis)
    booms.append(boom40)
    boom41 = boom.Boom(41, coordinates[41], 0.0, neutral_axis)
    booms.append(boom41)
    boom42 = boom.Boom(42, coordinates[42], 0.0, neutral_axis)
    booms.append(boom42)


    # CREATE EDGES INSTANCES AND PUT THEM IN EDGE LIST
    edge_list = []
    edge10 = edges.Edge([1, 0], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge10)
    edge21 = edges.Edge([2, 1], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge21)
    edge32 = edges.Edge([3, 2], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge32)
    edge43 = edges.Edge([4, 3], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge43)
    edge54 = edges.Edge([5, 4], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge54)
    edge65 = edges.Edge([6, 5], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge65)
    edge76 = edges.Edge([7, 6], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge76)
    edge87 = edges.Edge([8, 7], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge87)
    edge98 = edges.Edge([9, 8], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge98)
    edge109 = edges.Edge([10, 9], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge109)
    edge1110 = edges.Edge([11, 10], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge1110)
    edge1211 = edges.Edge([12, 11], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge1211)
    edge1312 = edges.Edge([13, 12], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge1312)
    edge1413 = edges.Edge([14, 13], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge1413)
    edge1514 = edges.Edge([15, 14], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge1514)

    # booms on semicircle
    edge1615 = edges.Edge([16, 15], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge1615)
    edge1716 = edges.Edge([17, 16], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge1716)
    edge1817 = edges.Edge([18, 17], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge1817)
    edge1918 = edges.Edge([19, 18], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge1918)
    edge2019 = edges.Edge([20, 19], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge2019)
    edge2120 = edges.Edge([21, 20], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge2120)
    edge2221 = edges.Edge([22, 21], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge2221)
```



```python
    edge2322 = edges.Edge([23, 22], 1.1*10**(-3), 44.179*10**(-3))
    edge_list.append(edge2322)

    # booms on lower spar
    edge2423 = edges.Edge([24, 23], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge2423)
    edge2524 = edges.Edge([25, 24], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge2524)
    edge2625 = edges.Edge([26, 25], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge2625)
    edge2726 = edges.Edge([27, 26], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge2726)
    edge2827 = edges.Edge([28, 27], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge2827)
    edge2928 = edges.Edge([29, 28], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge2928)
    edge3029 = edges.Edge([30, 29], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3029)
    edge3130 = edges.Edge([31, 30], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3130)
    edge3231 = edges.Edge([32, 31], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3231)
    edge3332 = edges.Edge([33, 32], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3332)
    edge3433 = edges.Edge([34, 33], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3433)
    edge3534 = edges.Edge([35, 34], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3534)
    edge3635 = edges.Edge([36, 35], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3635)
    edge3736 = edges.Edge([37, 36], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3736)
    edge3837 = edges.Edge([38, 37], 1.1*10**(-3), 56.103 * 0.5*10**(-3))
    edge_list.append(edge3837)
    edge038 = edges.Edge([0, 38], 1.1*10**(-3), 56.103*10**(-3))
    edge_list.append(edge038)

    # booms on the vertical spar
    edge3915 = edges.Edge([39, 15], 2.9*10**(-3), 45*10**(-3))
    edge_list.append(edge3915)
    edge4039 = edges.Edge([40, 39], 2.9*10**(-3), 45*10**(-3))
    edge_list.append(edge4039)
    edge4140 = edges.Edge([41, 40], 2.9*10**(-3), 45*10**(-3))
    edge_list.append(edge4140)
    edge4241 = edges.Edge([42, 41], 2.9*10**(-3), 45*10**(-3))
    edge_list.append(edge4241)
    edge2342 = edges.Edge([23, 42], 2.9*10**(-3), 45*10**(-3))
    edge_list.append(edge2342)

    # CREATE INSTANCE OF AILERON GEOMETRY WITH ALL THE BOOMS AND EDGES
    aileron_geometry = geometry.Geometry(43, booms, edge_list, [19880.391*10**(-6), 36225*10**(-6)], 28 * 10**9)
    aileron_geometry.construct_geometry()
    aileron_geometry.calc_centroid()
    aileron_geometry.cells = [[edge038, edge3837, edge3736, edge3635, edge3534, edge3433, edge3332, edge3231, edge3130,
                               edge3029, edge2928, edge2827, edge2726, edge2625, edge2524, edge2423, edge2342, edge4241,
                               edge4140, edge4039, edge3915, edge1514, edge1413, edge1312, edge1211, edge1110, edge109,
                               edge98, edge87, edge76, edge65, edge54, edge43, edge32, edge21, edge10],
                              [edge2019, edge1918, edge1817, edge1716, edge1615, edge3915, edge4039, edge4140, edge4241,
                               edge2342, edge2322, edge2221, edge2120]]

    # calculate areas and distances to centroid of all booms
    for element in booms:
        element.calculate_area(aileron_geometry)
    for boom_element in booms:
        boom_element.calc_y_dist(aileron_geometry)
        boom_element.calc_z_dist(aileron_geometry)
    aileron_geometry.get_areas()

    # calculate moments of inertia
    aileron_geometry.moment_inertia_Izz()
    aileron_geometry.moment_inertia_Iyy()

    # PLOT AND PRINT GEOMETRICAL PROPERTIES
    aileron_geometry.plot_edges()
    for it, el in enumerate(booms):
            print('area of boom ', it, ' : ', aileron_geometry.boom_areas[it], '[mm^2]')
    print('centroid position : ', aileron_geometry.centroid)
    print('z moment of inertia : ', aileron_geometry.Izz, ' [mm^4]')
    print('y moment of inertia : ', aileron_geometry.Iyy, ' [mm^4]')

    # GET THE LIST OF FORCES AND MOMENTS
    file_name = "Loads.txt"
    x_i_array = helpers.get_array_x_i(file_name)
    Mx_array = helpers.get_array_Mx_i(file_name)
    My_array = helpers.get_array_My_i(file_name)
    Mz_array = helpers.get_array_Mz_i(file_name)
    Sz_array = helpers.get_array_Sz_i(file_name)
    Sy_array = helpers.get_array_Sy_i(file_name)

    # create a matrix of normal stresses
    stress_matrix = np.zeros((43, 101))
    for j, location in enumerate(x_i_array):
```



```python
        for i, boom_member in enumerate(aileron_geometry.booms):
            boom_member.calc_bending_stress(Mz_array[j], My_array[j], aileron_geometry)
            stress_matrix[i][j] = boom_member.bending_stress

    # find maximum stress on each boom along the span
    max_stress_matrix = np.amax(stress_matrix, axis=1)
    print(max_stress_matrix)

    # set up matrix of shear stresses
    stress_matrix_shear = np.zeros((len(aileron_geometry.edges), 101))
    # set up lists of twist rates and thetas
    twist_rate_list = []
    thetas_list = []
    section_numbers = 100
    step = 2.771 / section_numbers
    thetas_list.append(0.453786)\
    # create file to store the list of thetas
    file = open("thetas_list.txt", "w")

    for i, x_i in enumerate(x_i_array):
        # create new instance of section with new location
        aileron_section = DiscreteSection.DiscreteSection(neutral_axis, aileron_geometry)
        # calculate shear flows due to pure shear and torque
        aileron_section.calc_total_shear_flow(Sz_array[i], Sy_array[i], Mx_array[i], edge2342)
        # calculate shear stress due to total shear flows and insert in the shear stress matrix
        aileron_section.calc_shear_stress()
        for n1, edge_ex in enumerate(aileron_geometry.edges):
            stress_matrix_shear[n1][i] = edge_ex.shear_stress
        # append the twist rate (computed at the same time as torque shear flow) in the twist rate list
        twist_rate_list.append(aileron_section.twist_rate)
        # calculate theta with finite differences, append to the list and copy to the txt file
        theta = twist_rate_list[i-1] * step + thetas_list[i-1]
        thetas_list.append(theta)
        file.write(str(float(theta)) + '\n')
    file.close()

    # find the maximum shear stress on each rib
    print('the maximum shear stress in rib A : ', np.max(stress_matrix_shear[:, 97]))
    print('the maximum shear stress in rib B : ', np.max(stress_matrix_shear[:, 51]))
    print('the maximum shear stress in rib C : ', np.max(stress_matrix_shear[:, 41]))
    print('the maximum shear stress in rib D : ', np.max(stress_matrix_shear[:, 18]))

    # find the maximum normal stress on each rib
    print('the maximum normal stress in rib A : ', np.max(stress_matrix[:, 97]))
    print('the maximum normal stress in rib A : ', np.max(stress_matrix[:, 51]))
    print('the maximum normal stress in rib A : ', np.max(stress_matrix[:, 41]))
    print('the maximum normal stress in rib A : ', np.max(stress_matrix[:, 18]))

    # plot twist angle along the span
    plt.plot(x_i_array, thetas_list[:-1])
    plt.show()

main()
```