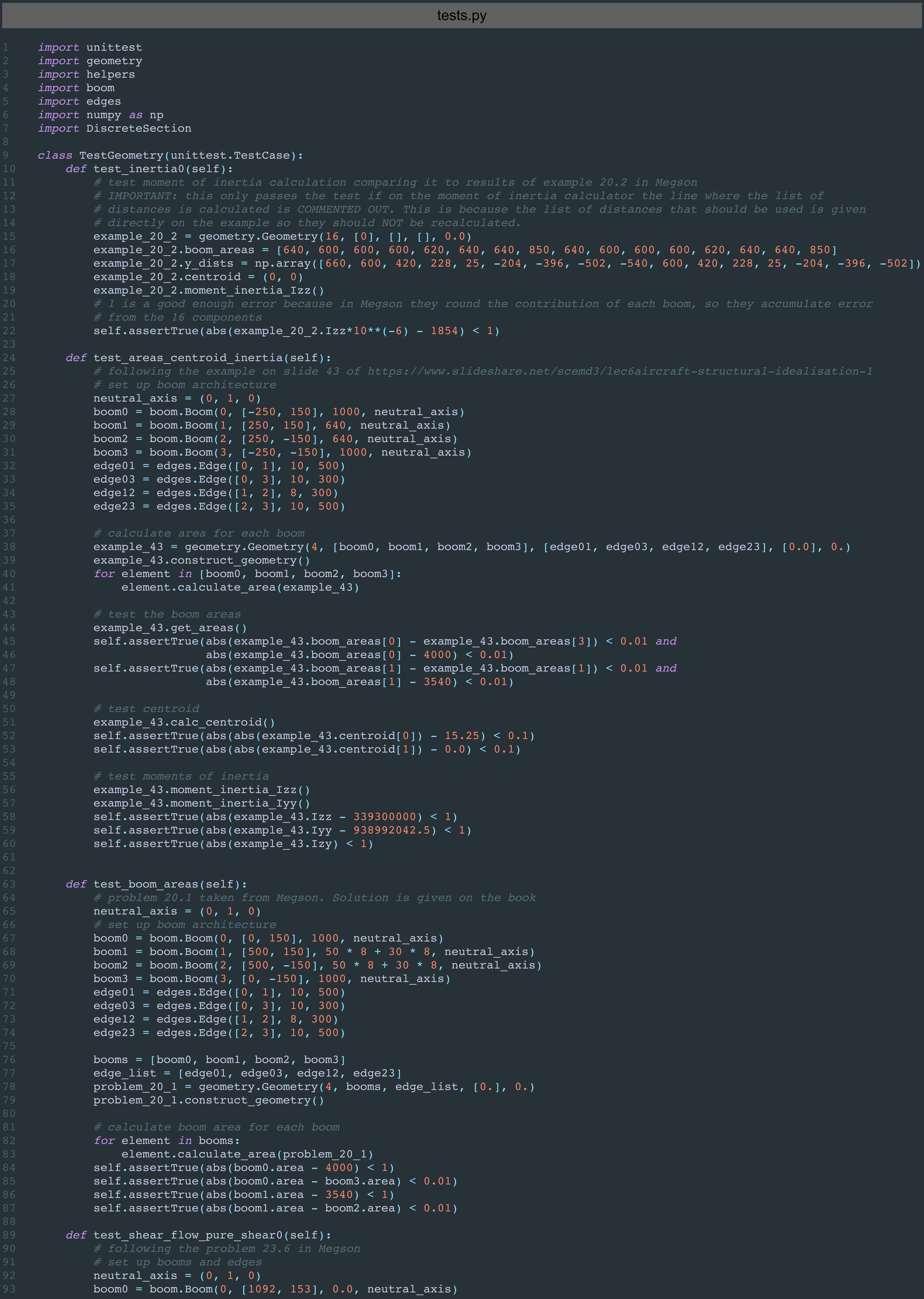
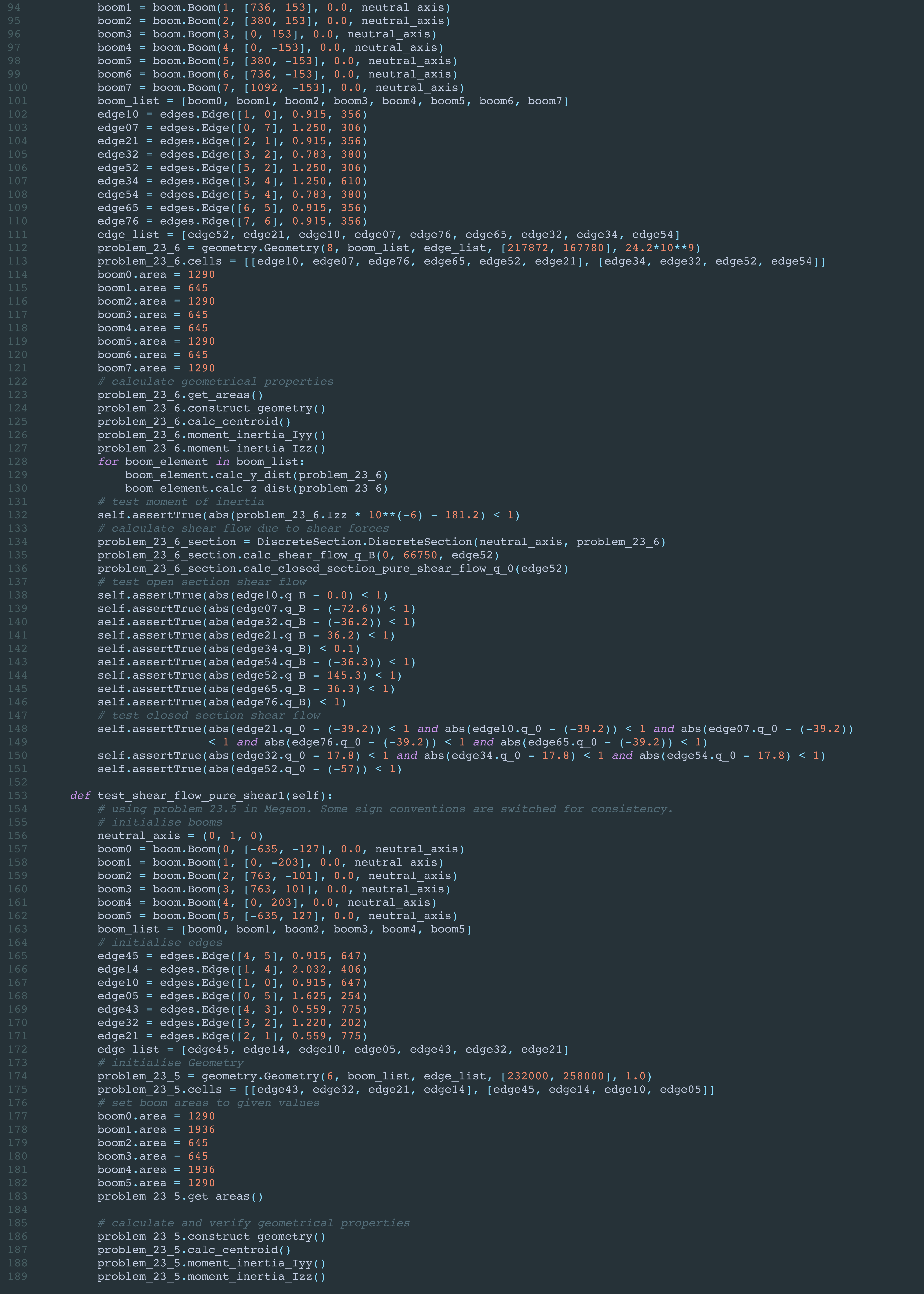
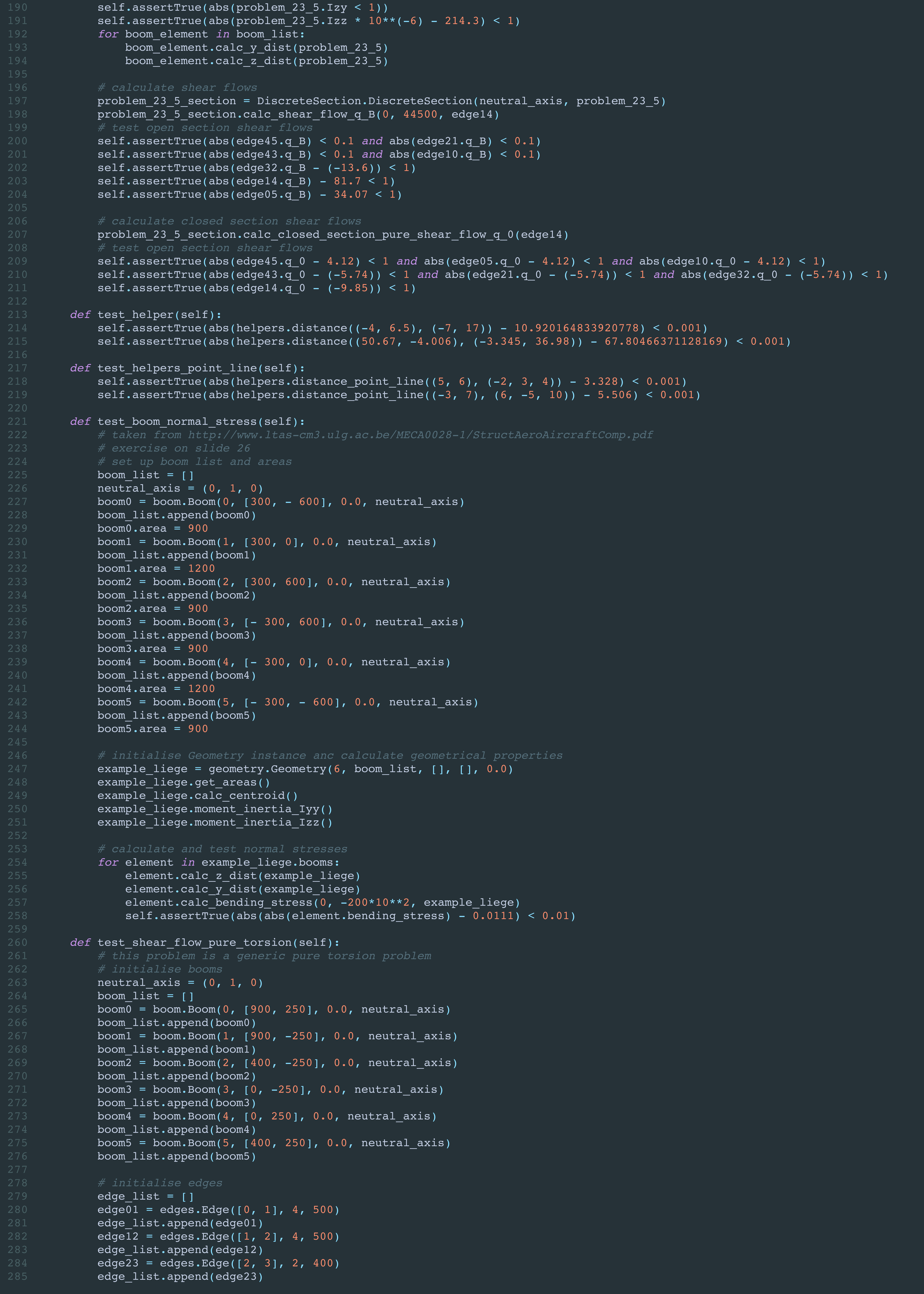

tests.py

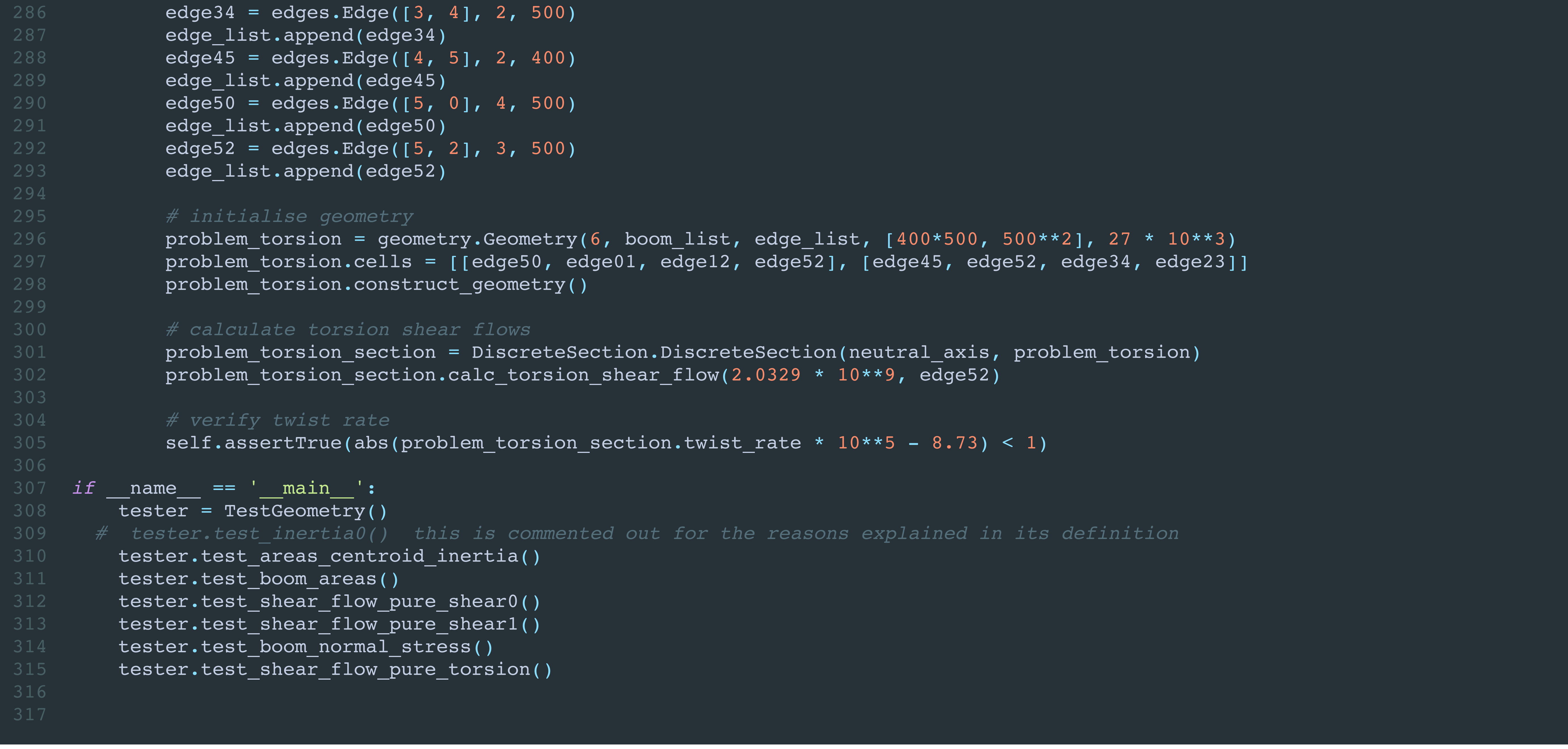
```python
import unittest
import geometry
import helpers
import boom
import edges
import numpy as np
import DiscreteSection

class TestGeometry(unittest.TestCase):
    def test_inertia0(self):
        # test moment of inertia calculation comparing it to results of example 20.2 in Megson
        # IMPORTANT: this only passes the test if on the moment of inertia calculator the line where the list of
        # distances is calculated is COMMENTED OUT. This is because the list of distances that should be used is given
        # directly on the example so they should NOT be recalculated.
        example_20_2 = geometry.Geometry(16, [0], [], [], 0.0)
        example_20_2.boom_areas = [640, 600, 600, 600, 620, 640, 640, 850, 640, 600, 600, 600, 620, 640, 640, 850]
        example_20_2.y_dists = np.array([660, 600, 420, 228, 25, -204, -396, -502, -540, 600, 420, 228, 25, -204, -396, -502])
        example_20_2.centroid = (0, 0)
        example_20_2.moment_inertia_Izz()
        # 1 is a good enough error because in Megson they round the contribution of each boom, so they accumulate error
        # from the 16 components
        self.assertTrue(abs(example_20_2.Izz*10**(-6) - 1854) < 1)

    def test_areas_centroid_inertia(self):
        # following the example on slide 43 of https://www.slideshare.net/scemd3/lec6aircraft-structural-idealisation-1
        # set up boom architecture
        neutral_axis = (0, 1, 0)
        boom0 = boom.Boom(0, [-250, 150], 1000, neutral_axis)
        boom1 = boom.Boom(1, [250, 150], 640, neutral_axis)
        boom2 = boom.Boom(2, [250, -150], 640, neutral_axis)
        boom3 = boom.Boom(3, [-250, -150], 1000, neutral_axis)
        edge01 = edges.Edge([0, 1], 10, 500)
        edge03 = edges.Edge([0, 3], 10, 300)
        edge12 = edges.Edge([1, 2], 8, 300)
        edge23 = edges.Edge([2, 3], 10, 500)

        # calculate area for each boom
        example_43 = geometry.Geometry(4, [boom0, boom1, boom2, boom3], [edge01, edge03, edge12, edge23], [0.0], 0.)
        example_43.construct_geometry()
        for element in [boom0, boom1, boom2, boom3]:
            element.calculate_area(example_43)

        # test the boom areas
        example_43.get_areas()
        self.assertTrue(abs(example_43.boom_areas[0] - example_43.boom_areas[3]) < 0.01 and
                        abs(example_43.boom_areas[0] - 4000) < 0.01)
        self.assertTrue(abs(example_43.boom_areas[1] - example_43.boom_areas[1]) < 0.01 and
                        abs(example_43.boom_areas[1] - 3540) < 0.01)

        # test centroid
        example_43.calc_centroid()
        self.assertTrue(abs(abs(example_43.centroid[0]) - 15.25) < 0.1)
        self.assertTrue(abs(abs(example_43.centroid[1]) - 0.0) < 0.1)

        # test moments of inertia
        example_43.moment_inertia_Izz()
        example_43.moment_inertia_Iyy()
        self.assertTrue(abs(example_43.Izz - 339300000) < 1)
        self.assertTrue(abs(example_43.Iyy - 938992042.5) < 1)
        self.assertTrue(abs(example_43.Izy) < 1)


    def test_boom_areas(self):
        # problem 20.1 taken from Megson. Solution is given on the book
        neutral_axis = (0, 1, 0)
        # set up boom architecture
        boom0 = boom.Boom(0, [0, 150], 1000, neutral_axis)
        boom1 = boom.Boom(1, [500, 150], 50 * 8 + 30 * 8, neutral_axis)
        boom2 = boom.Boom(2, [500, -150], 50 * 8 + 30 * 8, neutral_axis)
        boom3 = boom.Boom(3, [0, -150], 1000, neutral_axis)
        edge01 = edges.Edge([0, 1], 10, 500)
        edge03 = edges.Edge([0, 3], 10, 300)
        edge12 = edges.Edge([1, 2], 8, 300)
        edge23 = edges.Edge([2, 3], 10, 500)

        booms = [boom0, boom1, boom2, boom3]
        edge_list = [edge01, edge03, edge12, edge23]
        problem_20_1 = geometry.Geometry(4, booms, edge_list, [0.], 0.)
        problem_20_1.construct_geometry()

        # calculate boom area for each boom
        for element in booms:
            element.calculate_area(problem_20_1)
        self.assertTrue(abs(boom0.area - 4000) < 1)
        self.assertTrue(abs(boom0.area - boom3.area) < 0.01)
        self.assertTrue(abs(boom1.area - 3540) < 1)
        self.assertTrue(abs(boom1.area - boom2.area) < 0.01)

    def test_shear_flow_pure_shear0(self):
        # following the problem 23.6 in Megson
        # set up booms and edges
        neutral_axis = (0, 1, 0)
        boom0 = boom.Boom(0, [1092, 153], 0.0, neutral_axis)
```



```python
        boom1 = boom.Boom(1, [736, 153], 0.0, neutral_axis)
        boom2 = boom.Boom(2, [380, 153], 0.0, neutral_axis)
        boom3 = boom.Boom(3, [0, 153], 0.0, neutral_axis)
        boom4 = boom.Boom(4, [0, -153], 0.0, neutral_axis)
        boom5 = boom.Boom(5, [380, -153], 0.0, neutral_axis)
        boom6 = boom.Boom(6, [736, -153], 0.0, neutral_axis)
        boom7 = boom.Boom(7, [1092, -153], 0.0, neutral_axis)
        boom_list = [boom0, boom1, boom2, boom3, boom4, boom5, boom6, boom7]
        edge10 = edges.Edge([1, 0], 0.915, 356)
        edge07 = edges.Edge([0, 7], 1.250, 306)
        edge21 = edges.Edge([2, 1], 0.915, 356)
        edge32 = edges.Edge([3, 2], 0.783, 380)
        edge52 = edges.Edge([5, 2], 1.250, 306)
        edge34 = edges.Edge([3, 4], 1.250, 610)
        edge54 = edges.Edge([5, 4], 0.783, 380)
        edge65 = edges.Edge([6, 5], 0.915, 356)
        edge76 = edges.Edge([7, 6], 0.915, 356)
        edge_list = [edge52, edge21, edge10, edge07, edge76, edge65, edge32, edge34, edge54]
        problem_23_6 = geometry.Geometry(8, boom_list, edge_list, [217872, 167780], 24.2*10**9)
        problem_23_6.cells = [[edge10, edge07, edge76, edge65, edge52, edge21], [edge34, edge32, edge52, edge54]]
        boom0.area = 1290
        boom1.area = 645
        boom2.area = 1290
        boom3.area = 645
        boom4.area = 645
        boom5.area = 1290
        boom6.area = 645
        boom7.area = 1290
        # calculate geometrical properties
        problem_23_6.get_areas()
        problem_23_6.construct_geometry()
        problem_23_6.calc_centroid()
        problem_23_6.moment_inertia_Iyy()
        problem_23_6.moment_inertia_Izz()
        for boom_element in boom_list:
            boom_element.calc_y_dist(problem_23_6)
            boom_element.calc_z_dist(problem_23_6)
        # test moment of inertia
        self.assertTrue(abs(problem_23_6.Izz * 10**(-6) - 181.2) < 1)
        # calculate shear flow due to shear forces
        problem_23_6_section = DiscreteSection.DiscreteSection(neutral_axis, problem_23_6)
        problem_23_6_section.calc_shear_flow_q_B(0, 66750, edge52)
        problem_23_6_section.calc_closed_section_pure_shear_flow_q_0(edge52)
        # test open section shear flow
        self.assertTrue(abs(edge10.q_B - 0.0) < 1)
        self.assertTrue(abs(edge07.q_B - (-72.6)) < 1)
        self.assertTrue(abs(edge32.q_B - (-36.2)) < 1)
        self.assertTrue(abs(edge21.q_B - 36.2) < 1)
        self.assertTrue(abs(edge34.q_B) < 0.1)
        self.assertTrue(abs(edge54.q_B - (-36.3)) < 1)
        self.assertTrue(abs(edge52.q_B - 145.3) < 1)
        self.assertTrue(abs(edge65.q_B - 36.3) < 1)
        self.assertTrue(abs(edge76.q_B) < 1)
        # test closed section shear flow
        self.assertTrue(abs(edge21.q_0 - (-39.2)) < 1 and abs(edge10.q_0 - (-39.2)) < 1 and abs(edge07.q_0 - (-39.2))
                        < 1 and abs(edge76.q_0 - (-39.2)) < 1 and abs(edge65.q_0 - (-39.2)) < 1)
        self.assertTrue(abs(edge32.q_0 - 17.8) < 1 and abs(edge34.q_0 - 17.8) < 1 and abs(edge54.q_0 - 17.8) < 1)
        self.assertTrue(abs(edge52.q_0 - (-57)) < 1)

    def test_shear_flow_pure_shear1(self):
        # using problem 23.5 in Megson. Some sign conventions are switched for consistency.
        # initialise booms
        neutral_axis = (0, 1, 0)
        boom0 = boom.Boom(0, [-635, -127], 0.0, neutral_axis)
        boom1 = boom.Boom(1, [0, -203], 0.0, neutral_axis)
        boom2 = boom.Boom(2, [763, -101], 0.0, neutral_axis)
        boom3 = boom.Boom(3, [763, 101], 0.0, neutral_axis)
        boom4 = boom.Boom(4, [0, 203], 0.0, neutral_axis)
        boom5 = boom.Boom(5, [-635, 127], 0.0, neutral_axis)
        boom_list = [boom0, boom1, boom2, boom3, boom4, boom5]
        # initialise edges
        edge45 = edges.Edge([4, 5], 0.915, 647)
        edge14 = edges.Edge([1, 4], 2.032, 406)
        edge10 = edges.Edge([1, 0], 0.915, 647)
        edge05 = edges.Edge([0, 5], 1.625, 254)
        edge43 = edges.Edge([4, 3], 0.559, 775)
        edge32 = edges.Edge([3, 2], 1.220, 202)
        edge21 = edges.Edge([2, 1], 0.559, 775)
        edge_list = [edge45, edge14, edge10, edge05, edge43, edge32, edge21]
        # initialise Geometry
        problem_23_5 = geometry.Geometry(6, boom_list, edge_list, [232000, 258000], 1.0)
        problem_23_5.cells = [[edge43, edge32, edge21, edge14], [edge45, edge14, edge10, edge05]]
        # set boom areas to given values
        boom0.area = 1290
        boom1.area = 1936
        boom2.area = 645
        boom3.area = 645
        boom4.area = 1936
        boom5.area = 1290
        problem_23_5.get_areas()

        # calculate and verify geometrical properties
        problem_23_5.construct_geometry()
        problem_23_5.calc_centroid()
        problem_23_5.moment_inertia_Iyy()
        problem_23_5.moment_inertia_Izz()
```



```python
        self.assertTrue(abs(problem_23_5.Izy < 1))
        self.assertTrue(abs(problem_23_5.Izz * 10**(-6) - 214.3) < 1)
        for boom_element in boom_list:
            boom_element.calc_y_dist(problem_23_5)
            boom_element.calc_z_dist(problem_23_5)

        # calculate shear flows
        problem_23_5_section = DiscreteSection.DiscreteSection(neutral_axis, problem_23_5)
        problem_23_5_section.calc_shear_flow_q_B(0, 44500, edge14)
        # test open section shear flows
        self.assertTrue(abs(edge45.q_B) < 0.1 and abs(edge21.q_B) < 0.1)
        self.assertTrue(abs(edge43.q_B) < 0.1 and abs(edge10.q_B) < 0.1)
        self.assertTrue(abs(edge32.q_B - (-13.6)) < 1)
        self.assertTrue(abs(edge14.q_B) - 81.7 < 1)
        self.assertTrue(abs(edge05.q_B) - 34.07 < 1)

        # calculate closed section shear flows
        problem_23_5_section.calc_closed_section_pure_shear_flow_q_0(edge14)
        # test open section shear flows
        self.assertTrue(abs(edge45.q_0 - 4.12) < 1 and abs(edge05.q_0 - 4.12) < 1 and abs(edge10.q_0 - 4.12) < 1)
        self.assertTrue(abs(edge43.q_0 - (-5.74)) < 1 and abs(edge21.q_0 - (-5.74)) < 1 and abs(edge32.q_0 - (-5.74)) < 1)
        self.assertTrue(abs(edge14.q_0 - (-9.85)) < 1)

    def test_helper(self):
        self.assertTrue(abs(helpers.distance((-4, 6.5), (-7, 17)) - 10.920164833920778) < 0.001)
        self.assertTrue(abs(helpers.distance((50.67, -4.006), (-3.345, 36.98)) - 67.80466371128169) < 0.001)

    def test_helpers_point_line(self):
        self.assertTrue(abs(helpers.distance_point_line((5, 6), (-2, 3, 4)) - 3.328) < 0.001)
        self.assertTrue(abs(helpers.distance_point_line((-3, 7), (6, -5, 10)) - 5.506) < 0.001)

    def test_boom_normal_stress(self):
        # taken from http://www.ltas-cm3.ulg.ac.be/MECA0028-1/StructAeroAircraftComp.pdf
        # exercise on slide 26
        # set up boom list and areas
        boom_list = []
        neutral_axis = (0, 1, 0)
        boom0 = boom.Boom(0, [300, - 600], 0.0, neutral_axis)
        boom_list.append(boom0)
        boom0.area = 900
        boom1 = boom.Boom(1, [300, 0], 0.0, neutral_axis)
        boom_list.append(boom1)
        boom1.area = 1200
        boom2 = boom.Boom(2, [300, 600], 0.0, neutral_axis)
        boom_list.append(boom2)
        boom2.area = 900
        boom3 = boom.Boom(3, [- 300, 600], 0.0, neutral_axis)
        boom_list.append(boom3)
        boom3.area = 900
        boom4 = boom.Boom(4, [- 300, 0], 0.0, neutral_axis)
        boom_list.append(boom4)
        boom4.area = 1200
        boom5 = boom.Boom(5, [- 300, - 600], 0.0, neutral_axis)
        boom_list.append(boom5)
        boom5.area = 900

        # initialise Geometry instance anc calculate geometrical properties
        example_liege = geometry.Geometry(6, boom_list, [], [], 0.0)
        example_liege.get_areas()
        example_liege.calc_centroid()
        example_liege.moment_inertia_Iyy()
        example_liege.moment_inertia_Izz()

        # calculate and test normal stresses
        for element in example_liege.booms:
            element.calc_z_dist(example_liege)
            element.calc_y_dist(example_liege)
            element.calc_bending_stress(0, -200*10**2, example_liege)
            self.assertTrue(abs(abs(element.bending_stress) - 0.0111) < 0.01)

    def test_shear_flow_pure_torsion(self):
        # this problem is a generic pure torsion problem
        # initialise booms
        neutral_axis = (0, 1, 0)
        boom_list = []
        boom0 = boom.Boom(0, [900, 250], 0.0, neutral_axis)
        boom_list.append(boom0)
        boom1 = boom.Boom(1, [900, -250], 0.0, neutral_axis)
        boom_list.append(boom1)
        boom2 = boom.Boom(2, [400, -250], 0.0, neutral_axis)
        boom_list.append(boom2)
        boom3 = boom.Boom(3, [0, -250], 0.0, neutral_axis)
        boom_list.append(boom3)
        boom4 = boom.Boom(4, [0, 250], 0.0, neutral_axis)
        boom_list.append(boom4)
        boom5 = boom.Boom(5, [400, 250], 0.0, neutral_axis)
        boom_list.append(boom5)

        # initialise edges
        edge_list = []
        edge01 = edges.Edge([0, 1], 4, 500)
        edge_list.append(edge01)
        edge12 = edges.Edge([1, 2], 4, 500)
        edge_list.append(edge12)
        edge23 = edges.Edge([2, 3], 2, 400)
        edge_list.append(edge23)
```



```
        edge34 = edges.Edge([3, 4], 2, 500)
        edge_list.append(edge34)
        edge45 = edges.Edge([4, 5], 2, 400)
        edge_list.append(edge45)
        edge50 = edges.Edge([5, 0], 4, 500)
        edge_list.append(edge50)
        edge52 = edges.Edge([5, 2], 3, 500)
        edge_list.append(edge52)

        # initialise geometry
        problem_torsion = geometry.Geometry(6, boom_list, edge_list, [400*500, 500**2], 27 * 10**3)
        problem_torsion.cells = [[edge50, edge01, edge12, edge52], [edge45, edge52, edge34, edge23]]
        problem_torsion.construct_geometry()

        # calculate torsion shear flows
        problem_torsion_section = DiscreteSection.DiscreteSection(neutral_axis, problem_torsion)
        problem_torsion_section.calc_torsion_shear_flow(2.0329 * 10**9, edge52)

        # verify twist rate
        self.assertTrue(abs(problem_torsion_section.twist_rate * 10**5 - 8.73) < 1)

if __name__ == '__main__':
    tester = TestGeometry()
  #  tester.test_inertia0()  this is commented out for the reasons explained in its definition
    tester.test_areas_centroid_inertia()
    tester.test_boom_areas()
    tester.test_shear_flow_pure_shear0()
    tester.test_shear_flow_pure_shear1()
    tester.test_boom_normal_stress()
    tester.test_shear_flow_pure_torsion()


```