Summer Fellowship Report

On

Osdag

Submitted by

Tanmay Kalla

Under the guidance of

Prof. Siddhartha Ghosh
Civil Engineering Department
IIT Bombay
&
Mentors - Danish Ansari, Ajmal Babu MS
FOSSEE, Osdag
IIT Bombay

July 12, 2019

Acknowledgment

I would like to thank the FOSSEE project from IIT Bombay for giving me an opportunity to do internship with Osdag. The internship opportunity was a great chance for me to learn and develop myself professionally. It helped me to enhance my knowledge in structural steel design,Object Oriented Programming,python data structures and software testing. I feel grateful to have met so many wonderful people and professionals who guided me through this internship period.

I would like to specially acknowledge Prof. Siddhartha Ghosh with my deepest gratitude who in spite of being busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed research lab (SSRR lab).

I would also like to acknowledge the members of the Osdag team; Danish Ansari (Project Research Assistant), Ajmal Babu MS (Project Research Engineer), and Sourabh Das (Project Research associate) for their careful and precious guidance which was extremely valuable for my both theoretical and practical study.

Contents

Chapter 1

Introduction to Osdag Internship

Osdag internship is provided under the FOSSEE project. FOSSEE project promotes the use of FLOSS (Free Libre and Open Source Software) tools to improve quality of education in our country. FOSSEE encourages the use of FOSS tools through various activities to ensure commercial (paid) softwares are replaced by equivalent FOSS tools.

The FOSSEE project is a part of the National Mission on Education through Infrastructure and Communication Technology(ICT), Ministry of Human Resources and Development, Government of India.

Osdag is one such open source software which comes under the FOSSEE project.

1.1 What is Osdag ?

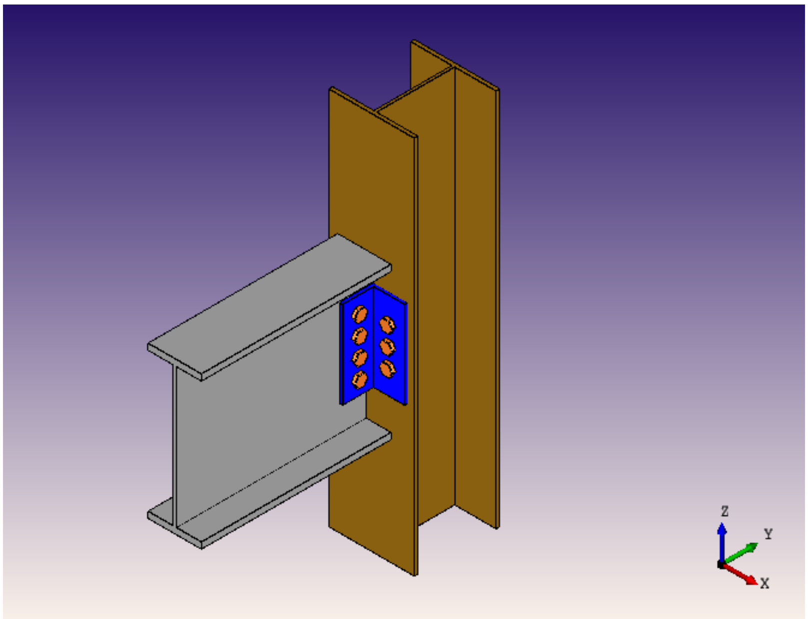
Osdag is a cross-platform free and open-source software for the design of steel structures, following the Indian standard IS

800:2007. It allows the users to design steel connections, members and systems using a graphical user interface. The interactive GUI provides a 3D visualisation of the designed component and creates images for construction/fabrication drawings.

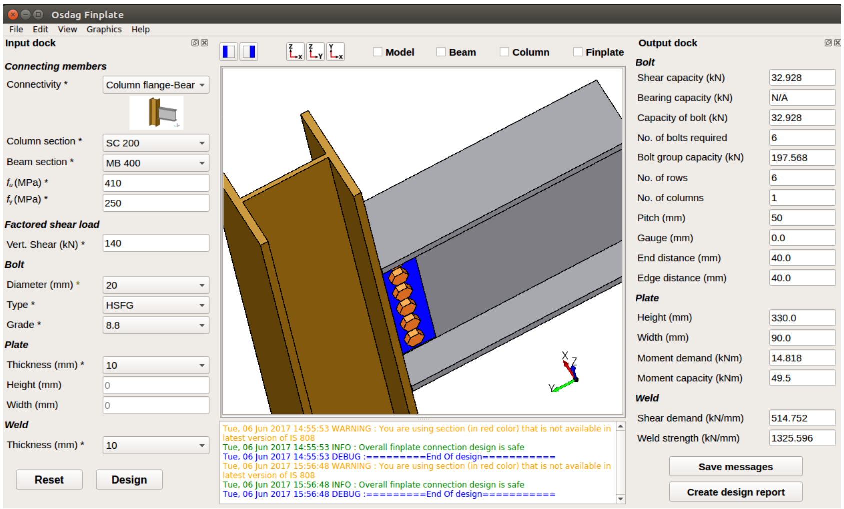
It is used for solving steel structures problems and to see how the connection will look after practical implementation. There are different modules available in Osdag with various connectivities.

Osdag provides various features such as:

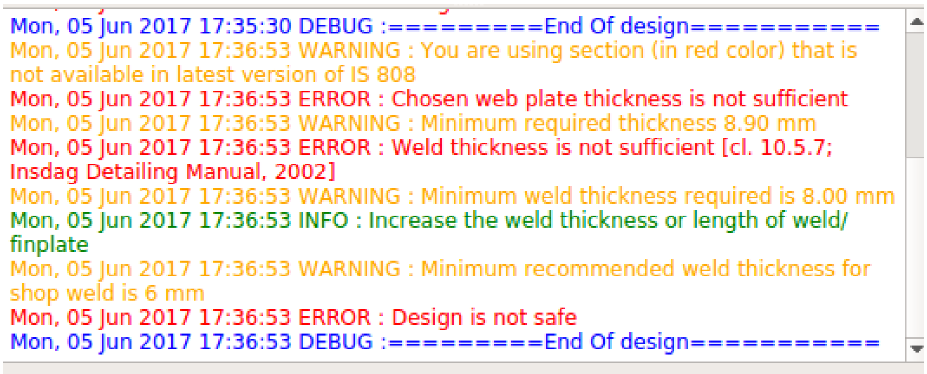
- An interactive window displaying a 3D CAD model, which provides a clear visualisation of the designed component.
- Creation of 3D CAD models that can be imported to generic CAD softwares.



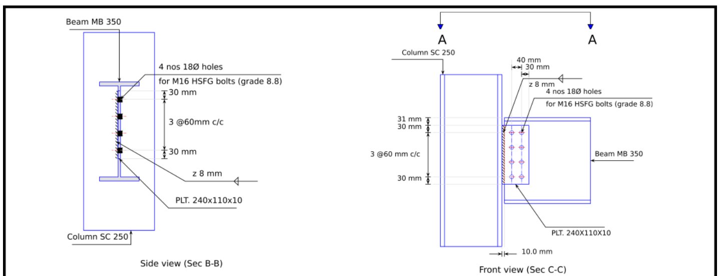
- User-friendly input and output docs, with text-validated fields grouped according to the design flow.



- A text window for message display, that also suggests necessary changes if a trial design is found unsafe.

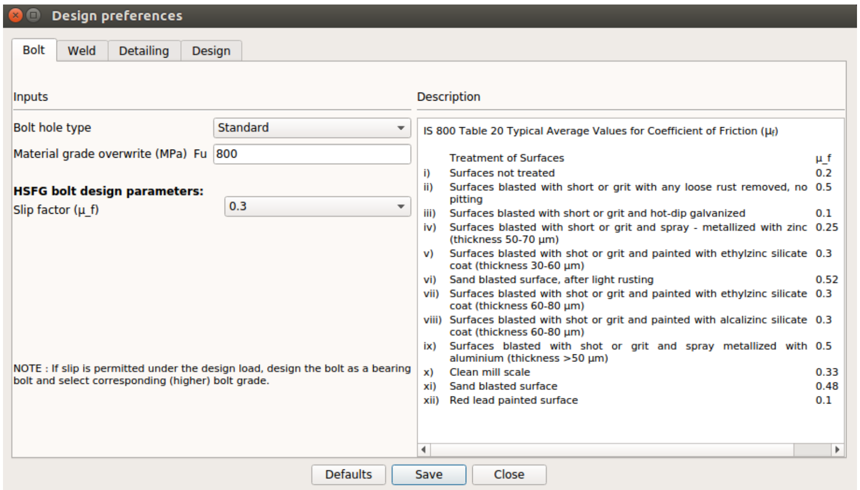


- Creation of a professional design report showing all necessary checks, design calculations as per IS 800:2007, and standard views of the designed component.



Design Check			
Check	Required	Provided	Remark
Bolt shear capacity(kN)		Vdsf = ((0.3*1*1.0*87.92)/(1.25)) = 21.1008 [cl. 10.4.3]	
Bolt bearing capacity(kN)		NA	
Bolt capacity (kN)		21.1008	Pass
No. of bolts	120/21.1008 = 5.7	7	Pass
No. of column(s)	≤ 2	2	
No. bolts per column		4	

- Creation of 2D vector (and raster) images that can be used in a design report or class assignment.
- Selection of design preferences, considering different construction and detailing aspects, using a design preference toolbox.



1.2 Who can use ?

Osdag is generally created for industry professionals but it also keep students in mind. As Osdag is funded by MHRD, Osdag team tries to manipulate software in such a way that it can be used by the students during their academics and to give them a better insight look in the subject.

Basically Osdag can be used by anyone starting from novice to professionals. It's simple and sober user interface makes it flexible and attractive than the other softwares. Also there are video tutorials to get started. The video tutorials of Osdag can be accessed here.

Chapter 2

Osdag Development

2.1 IS800:2007 Python Functions

IS 800:2007 is an Indian Standard code of practice for general construction in steel. Osdag uses it as a fundamental document to perform the design tasks. Code for section 3, section 5 and annex F of IS800 were added to the already existing python file. In which clause 3.7, 3.8, annex F and table 2, 3, 5 and 44 were added to the python code.

```
def null(self, b, tf, d, tw, t, D):

        cl_3_7_Table_2 = {"Compression_elements": {
                "outstanding_elements_compression_flange": {"rolled": b / tf, "welded": b / tf},
                "internal_elements_compression_flange": {"compression_due_to_bending": b / tf,
                                                          "axial_compression": b / tf},
                "web_of_a_channel": d / tw,
                "angle_compression_due_to_bending": {b / t, d / t},
                "single_angles_or_double_angles_with_seperated_elements_axial_compression": {b / t, d / t, (b + d) / t},
                "outstanding_leg_in_back_to_back_in_a_double_angle_member": d / t,
                "outstanding_leg_of_an_angle_with_its_back_in_cont_contact_with_another_component": d / t,
                "stem_of_tsection_rolled_or_cut_from_a_rolled_IorH_section": D / tf,
                "circular_hollow_tube_including_welded_tube_subjected_to": {"moment": D / t,
                                                                            "axial_compression": D / t},
                "web_of_an_I_or_H_section": {"general": d / tw, "axial_compression": d / tw}
            }
        }

def cl_3_7_3_class(self, cl_3_7_Table_2, e, r1):
        """ Gives class of cross sections using table 2
        Args:
             b - width of element (float)
             d - depth of web (float)
             t - thickness of element (float)
             tf - thickness of flange (float)
             tw - thickness of web (float)
             D - outer diameter of element (float)
             r1 - actual average stress/design compressive stress of web alone (float)
             r2 - actual average stress/design compressive stress of overall section (float)
             f_y - Yield stress of the plate material in MPa (float)
             e = sqrt(250/f_y)
        Return:
            Class - type of the cross section (string)
        Note:
            Reference: IS 800:2007, cl 3.7.2
        """
        if cl_3_7_Table_2[0]["outstanding_elements_compression_flange"]["rolled"] <= 9.4 * e:
            return "class1"
        elif 10.5 * e >= cl_3_7_Table_2[0]["outstanding_elements_compression_flange"]["rolled"] > 9.4 * e:
            return "class2"
        elif 15*e >= cl_3_7_Table_2[0]["outstanding_elements_compression_flange"]["rolled"] > 10.5 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["outstanding_elements_compression_flange"]["welded"] <= 8.4 * e:
            return "class1"
```

```
        elif  9.4 * e >= cl_3_7_Table_2[0]["outstanding_elements_compression_flange"]["welded"] > 8.4 * e:
            return "class2"
        elif 13.6 * e >= cl_3_7_Table_2[0]["outstanding_elements_compression_flange"]["welded"] > 9.4 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["internal_elements_compression_flange"]["compression_due_to_bending"] <= 29.3 * e:
            return "class1"
        elif 33.5 * e >= cl_3_7_Table_2[0]["internal_elements_compression_flange"]["compression_due_to_bending"]>29.3*e:
            return "class2"
        elif 42 * e >= cl_3_7_Table_2[0]["internal_elements_compression_flange"]["compression_due_to_bending"] > 33.5 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["internal_elements_compression_flange"]["axial_compression"] >= 42 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["web_of_a_channel"] <= 42 * e:
            return "class1 or class2 or class3"
        elif cl_3_7_Table_2[0]["angle_compression_due_to_bending"][0] <= 9.4 * e and \
                cl_3_7_Table_2[1]["angle_compression_due_to_bending"][0] <= 9.4 * e:
            return "class1"
        elif 10.5 * e >= cl_3_7_Table_2[0]["angle_compression_due_to_bending"][0] > 9.4 * e and \
                10.5 * e <= cl_3_7_Table_2[0]["angle_compression_due_to_bending"][1] > 9.4 * e:
            return "class2"
        elif 15.7 * e <= cl_3_7_Table_2[0]["angle_compression_due_to_bending"][0] > 10.5 * e and \
                15.7 * e <= cl_3_7_Table_2[0]["angle_compression_due_to_bending"][1] > 10.5 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["single_angles_or_double_angles_with_seperated_elements_axial_compression"][
            0] <= 15.7 * e and \
                cl_3_7_Table_2[0]["single_angles_or_double_angles_with_seperated_elements_axial_compression"][
                    1] <= 15.7 * e and \
                cl_3_7_Table_2[0]["single_angles_or_double_angles_with_seperated_elements_axial_compression"][
                    2] <= 25 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["outstanding_leg_in_back_to_back_in_a_double_angle_member"] <= 9.4 * e:
            return "class1"
        elif 10.5 * e >= cl_3_7_Table_2[0]["outstanding_leg_in_back_to_back_in_a_double_angle_member"] > 9.4 * e:
            return "class2"
        elif 15.7 * e >= cl_3_7_Table_2[0]["outstanding_leg_in_back_to_back_in_a_double_angle_member"] > 10.5 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["outstanding_leg_of_an_angle_with_its_back_in_cont_contact_with_another_component"] <= 9.4 * e:
            return "class1"
        elif 10.5 * e >= cl_3_7_Table_2[0]["outstanding_leg_of_an_angle_with_its_back_in_cont_contact_with_another_component"] > 9.4
        ↪    * e:
            return "class2"
        elif 15.7 * e >= cl_3_7_Table_2[0]["outstanding_leg_of_an_angle_with_its_back_in_cont_contact_with_another_component"] > 10.5
        ↪    * e:
            return "class3"
        elif cl_3_7_Table_2[0]["stem_of_tsection_rolled_or_cut_from_a_rolled_IorH_section"] <= 8.4 * e:
            return "class1"
        elif 9.4 * e >= cl_3_7_Table_2[0]["stem_of_tsection_rolled_or_cut_from_a_rolled_IorH_section"] > 8.4 * e:
            return "class2"
        elif 18.9 * e >= cl_3_7_Table_2[0]["stem_of_tsection_rolled_or_cut_from_a_rolled_IorH_section"] > 9.4 * e:
            return "class3"
        elif cl_3_7_Table_2[0]["circular_hollow_tube_including_welded_tube_subjected_to"][0] <= 42 * e * e:
            return "class1"
        elif 52 * e * e >= cl_3_7_Table_2[0]["circular_hollow_tube_including_welded_tube_subjected_to"][0] > 42 * e * e:
            return "class2"
        elif 146 * e * e >= cl_3_7_Table_2[0]["circular_hollow_tube_including_welded_tube_subjected_to"][0] > 52 * e * e:
            return "class3"
        elif cl_3_7_Table_2[0]["circular_hollow_tube_including_welded_tube_subjected_to"][1] <= 88 * e * e:
            return "class3"
        elif cl_3_7_Table_2[0]["web_of_an_I_or_H_section"]["general"] <= 84 * e / (1 + r1):
            return "class1"
        elif r1 < 0 and 84 * e / (1 + r1) > cl_3_7_Table_2[0]["web_of_an_I_or_H_section"]["general"] <= 105 * e / (1 + r1):
            return "class2"
        elif r1 > 0 and 84 * e / (1 + r1) > cl_3_7_Table_2[0]["web_of_an_I_or_H_section"]["general"] <= 105 * e / (1 + 1.5 * r1):
            return "class2"
        elif 105 * e / (1 + r1) > cl_3_7_Table_2[0]["web_of_an_I_or_H_section"]["general"] <= 126 * e / (1 + 2 * r1):
            return "class3"
        elif cl_3_7_Table_2[0]["web_of_an_I_or_H_section"]["axial_compression"] <= 42 * e:
            return "class3"


# Table 3 Maximum slendernesss ratio
    """ Table 5 gives the maximum effective slenderness ratio (KL/r) according to member type
           Slenderness ratio=KL/r
           KL:effective length of the member
           r:appropriate radius of gyration based on effective section
           Member types relating cases:
           case1:A member carrying compressive loads from dead loads and imposed loads
           case2:A tension member in which a reversal of direct stress occur dueto loads other than wind or seismic loads
           case3:A member subjected to compression forces resulting only from combination with wind/earthquake actions,
                 provided deformations does not adversely affect the stress in any part of the structure
           case4:Compression flange of a beam  against lateral torsional buckling
           case5:A member normally acting as tie in a roof truss or a bracing system not considered effective when
                 when subjected to possible reversal of stress into compression resulting from action of wind or earthquake
                  forces
           case6:Members always under tension(other than pre-tensioned members)"""

    cl_3_8_Table_3 = {"case1": 180,
                      "case2": 180,
                      "case3": 250,
                      "case4": 300,
                      "case5": 350,
                      "case6": 400}

AnnexF: Python code
"""    ANNEX  F       CONNECTIONS  """
```

```
    #Connnection classification
   """Connection classification using table 43 and 44"""
    def F_4_3_1 (d,da,db,dg,g,ta,tc,tf,tw,tp,la,lt,M,connection_type):
        """
        :param d:depth of beam (float)
        :param da:depth of the angle in mm (float)
        :param db:diameter of the bolt in mm (float)
        :param dg:centre to centre of outermost bolt of the end plate (float)
        :param g:gauge distance of bolt line (float)
        :param ta:thickness of the top angle in mm (float)
        :param tc:thickness of the web angle in mm (float)
        :param tf:thinkness of the flange T-stub connector in mm (float)
        :param tw:thickness of the web of the beam in the conncection in mm (float)
        :param tp:thickness of the end plate in mm(float)
        :param la:length of angle in mm (float)
        :param lt:length if the T-sub connector in mm(float)
        :param M: moment at the joint in KNm(float)
        :param connection_type: type of connection (str)
        :return:theta_r (float)
        """
        table_44 ={"A":{{"c1":1.91*10**4,"c2":1.3*10**11,"c3":2.7*10**17},(da**-2.4)*(tc**-1.81)*(g**0.15)},
                   "B":{{"c1":1.64*10**3,"c2":1.03*10**14,"c3":8.18*10**25},(da**-2.4)*(tc**-1.81)*(g**0.15)},
                   "C":{{"c1":2.24*10**(-1),"c2":1.86*10**4,"c3":3.23*10**8},(d**-1.287)*(tc**-0.415)*(ta**-1.128)*(la**-0.694)},
                   "D":{{"c1":1.63*10**3,"c2":7.25*10**14,"c3":3.31*10**23},(d**-1.5)*(ta**-0.5)*(la**-0.7)*(db**-1.1)},
                   "E":{{"c1":1.78*10**4,"c2":-9.55*10**16,"c3":5.54*10**29},(dg**-2.4)*(tp**-0.4)*(tf**-1.5)},
                   "F":{{"c1":2.6*10**2,"c2":-9.55*10**16,"c3":5.54*10**29},(dg**-2.4)*(tp**-0.4)},
                   "G":{{"c1":4.05*10**2,"c2":4.45*10**13,"c3":-2.03*10**23},(d**-1.5)*(tf**-0.5)*(lt**-0.7)*(db**-1.1)},
                   "H":{{"c1":3.87,"c2":2.71*10**5,"c3":6.06*10**11},(tp**-1.6)*(g**1.6)*(tw**-0.5)*(db**-2.3)}
                   }
        global theta_r
        theta_r=
         ↪    (table_44[connection_type][0]["c1"]*table_44[connection_type][1]*M)+(table_44[connection_type][0]["c2"]*(table_44[connection_type][1]*M)**3)+(tab
        return theta_r

    def connection_classification(theta_r,M,Mpb,theta_p):
        m1=M/Mpb
        o1=theta_r/theta_p
        if m1>1 or 3*m1+2*o1>5.4 or o1<0 or m1<0:
            return 0
        elif m1>=0.7 or m1>=2.5*o1:
            connection= "rigid connection"
        elif m1<=0.2 or m1<=0.5*o1:
            connection = "Flexible connection"
        else:
            connection="Semi rigid"
        return connection
```

Github Link Click Here

2.2 Component Class

Under restructuring the software, modular approach is used to make code more efficient. In this different modules of code are created in which various classes like input class, component class, material class etc.. are included. In the component class module, all steel components common to all design modules are included as classes like Bolt, Angle, Plate, Sections etc. and attributes relevant to each of them is included as instances. Methods to calculate the instances are also added. Python Code:

```
from material import Material
import sqlite3
from is800_2007 import IS800_2007

class Component(object):

    def __init__(self, material=Material()):
```

```
        self.material = material
        self.path_to_database = "../../databases/Intg_osdag.sqlite"


class Bolt(Component):

    def __init__(self, grade=0.0, diameter=0.0,thread_area=0, bolt_type="", length=0.0, material=Material()):
        self.grade = grade
        self.diameter = diameter
        self.shank_area = 3.14*0.25*diameter**2
        self.thread_area= thread_area
        self.bolt_type = bolt_type
        self.length = length
    #   bearing bolt
        self.shear_capacity = 0.0
        self.bearing_capacity = 0.0
        self.bolt_capacity = 0.0
    #   friction bolt
        self.is_friction_grip = True
        self.slip_resistance = 0
    #   both
        self.shear_in_bolt = 0
        self.tension_in_bolt = 0
        self.tension_capacity = 0
        self.combined_capacity_check = "safe"

        super(Bolt, self).__init__(material)

    def __repr__(self):
        repr = "Bolt\n"
        repr += "Diameter: {}\n".format(self.diameter)
        repr += "Type: {}\n".format(self.bolt_type)
        repr += "Grade: {}\n".format(self.grade)
        repr += "Length: {}".format(self.length)
        return repr



    def calculate_thread_area(self):
        self.thread_area = 0.78*self.shank_area
    # todo thread_area database
    """Friction bolt"""
    def calculate_slip_resistance(self, n_e ,mu_f):
        self.slip_resistance = IS800_2007.cl_10_4_3_bolt_slip_resistance(self.material.fub, self.shank_area, n_e, mu_f)

    # n_e and m_uf are plate attributes
    """bearing bolt """
    def calculate_bolt_shear_capacity(self):
        self.shear_capacity = IS800_2007.cl_10_3_3_bolt_shear_capacity(self.material.fub, self.shank_area, self.thread_area)

    def calculate_bolt_bearing_capacity(self ,t):
        self.bearing_capacity = IS800_2007.cl_10_3_4_bolt_bearing_capacity(self.material.fu, self.material.fub, t, self.diameter,BoltGroup.edge,BoltGroup.pitch)
    # use t , e , p from bolt group class

    def calculate_bolt_capacity(self):
        if self.is_friction_grip is False:
            self.bolt_capacity = min(self.bearing_capacity, self.shear_capacity)
    """same for both"""
    def calculate_tension_capacity(self, An):

        self.tension_capacity = min(0.9*self.material.fub*An, self.material.fyb*self.shank_area*1.25/1.1 )

    def calculate_combined_capacity(self, shear_capacity_of_friction_bolt):
        if self.is_friction_grip is False:
            if ((self.shear_in_bolt/1.25)/self.shear_capacity)**2 + ((self.tension_in_bolt/1.25)/self.tension_capacity)**2 <= 1:
                self.combined_capacity_check = "safe"
            else:
                self.combined_capacity_check = "unsafe"
        elif self.is_friction_grip is True:
            if ((self.shear_in_bolt/1.25)/shear_capacity_of_friction_bolt)**2 + ((self.tension_in_bolt/1.25)/self.tension_capacity)**2 <= 1:
                self.combined_capacity_check = "safe"
            else:
                self.combined_capacity_check = "unsafe"




class BoltGroup(Component):

    def __init__(self, bolt, no_rows, no_columns, gauge=0.0, pitch=0.0, end=0.0, edge=0.0, material=Material()):
        self.bolt = bolt
        self.no_rows = no_rows
        self.no_columns = no_columns
        self.no_of_bolts = no_rows * no_columns
        self.group_capacity = self.no_of_bolts * self.bolt.bolt_capacity
        self.gauge = gauge
        self.pitch = pitch
        self.end = end
        self.edge = edge
        super(BoltGroup, self).__init__(material)
    def __repr__(self):
        repr = "Bolt Group\n"
        repr += "no_of_bolts: {}\n".format(self.no_of_bolts)
```

```
        repr += "group_capacity {}\n".format(self.group_capacity)
        repr += "gauge {}\n".format(self.gauge)
        repr += "pitch {}\n".format(self.pitch)
        repr += "end {}\n".format(self.end)
        repr += "edge {}\n".format(self.edge)
        return repr

    def no_of_bolts_check(self, v_d, v_bolt):
        if self.no_of_bolts > v_d / v_bolt :
            return True
        if self.no_of_bolts > v_d / v_bolt :
            return False

    def min_pitch_check(self):
        if self.pitch >= IS800_2007.cl_10_2_2_min_spacing(self.pitch):
            return True
        else:
            return False

    def min_gauge_check(self):
        if self.gauge >= IS800_2007.cl_10_2_2_min_spacing(self.gauge):
            return True
        else:
            return False

    def max_pitch_check(self, plate):
        if self.pitch <= IS800_2007.cl_10_2_3_1_max_spacing(plate.thickness):
            return True
        else:
            return False


    def max_gauge_check(self, plate):
        if self.gauge <= IS800_2007.cl_10_2_3_1_max_spacing(plate.thickness):
            return True
        else:
            return False


    def max_pitch_check_2(self, plate, compression_or_tension):
        if self.pitch <= IS800_2007.cl_10_2_3_2_max_pitch_tension_compression(self.pitch, plate.thickness, compression_or_tension):
            return True
        else:
            return False

    def min_end_check(self, bolt_hole_type, edge_type):
        if self.end >= IS800_2007.cl_10_2_4_2_min_edge_end_dist(self.end, bolt_hole_type, edge_type):
            return True
        else:
            return False

    def min_edge_check(self,bolt_hole_type, edge_type):
        if self.edge >= IS800_2007.cl_10_2_4_2_min_edge_end_dist(self.end, bolt_hole_type, edge_type):
            return True
        else:
            return False

    def max_end_check(self, plate, f_y, corrosive_influences):
        if self.end <= IS800_2007.cl_10_2_4_3_max_edge_dist(plate.thickness, f_y, corrosive_influences):
            return True
        else:
            return False

    def max_edge_check(self, plate, f_y, corrosive_influences):
        if self.edge <= IS800_2007.cl_10_2_4_3_max_edge_dist(plate.thickness, f_y, corrosive_influences):
            return True
        else:
            return False


    def check_for_long_joints(self):
        l_j = (self.no_rows - 1) * self.pitch
        beta_lj = IS800_2007.cl_10_3_3_1_bolt_long_joint(self.bolt.diameter, l_j)
        return beta_lj

class Nut(Component):

    def __init__(self, diameter=0.0, material=Material()):
        self.diameter = diameter
        super(Nut, self).__init__(material)

    def __repr__(self):
        repr = "Nut\n"
        repr += "Diameter: {}".format(self.diameter)
        return repr

class Section(Component):

    def __init__(self, designation, material=Material()):
        self.designation = designation
        self.depth = 0.0
        self.flange_width = 0.0
        self.web_thickness = 0.0
        self.flange_thickness = 0.0
        self.root_radius = 0.0
```

```
        self.toe_radius = 0.0
        self.Ix = 0.0
        self.Iy = 0.0
        self.cx = 0.0
        self.cy = 0.0
        self.buckling_class = ""


        super(Section, self).__init__(material)

    def __repr__(self):
        repr = "Section\n"
        repr += "Designation: {}".format(self.designation)
        return repr

    def connect_to_database_update_other_attributes(self, table, designation):
        conn = sqlite3.connect(self.path_to_database)
        db_query = "SELECT D, B, tw, T, R1, R2 ,Iz ,Iy FROM " + table + " WHERE Designation = ?"
        cur = conn.cursor()
        cur.execute(db_query, (designation,))
        row = cur.fetchone()

        self.depth = row[0]
        self.flange_width = row[1]
        self.web_thickness = row[2]
        self.flange_thickness = row[3]
        self.root_radius = row[4]
        self.toe_radius = row[5]
        self.Ix = row[6]
        self.Iy = row[7]


        conn.close()





class Beam(Section):

    def __init__(self, designation, material=Material()):
        super(Beam, self).__init__(designation, material)
        self.connect_to_database_update_other_attributes("Beams", designation)



class Column(Section):

    def __init__(self, designation, material=Material()):
        super(Column, self).__init__(designation, material)
        self.connect_to_database_update_other_attributes("Columns", designation)





class Weld(Component):

    def __init__(self, type, size=0.0, length=0.0, eff_length = 0.0, material=Material()):
        self.type = type
        self.size = size
        self.length = length
        self.throat_size = size * 0.7
        self.eff_length = None
        self.shear_strength = None
        super(Weld, self).__init__(material)

    def __repr__(self):
        repr = "Weld\n"
        repr += "Type: {}\n".format(self.type)
        repr += "Size: {}\n".format(self.size)
        repr += "Length: {}".format(self.length)
        repr += "Throat_size: {}".format(self.throat_size)
        repr += "Eff_length: {}".format(self.eff_length)
        repr += "Shear_strength: {}".format(self.shear_strength)
        return repr

    def calculate_eff_length(self,available_length):
        self.eff_length = IS800_2007.cl_10_5_4_1_fillet_weld_effective_length(self.size, available_length)
        return self.eff_length

    def calculate_shear_strength(self,ultimate_stresses,fabrication):
        self.shear_strength = IS800_2007.cl_10_5_7_1_1_fillet_weld_design_stress(ultimate_stresses, fabrication)
        return self.shear_strength



class Plate(Component):

    def __init__(self, thickness, height, width, plate_type,material=Material()):
        self.thickness = thickness
        self.height = height
        self.width = width
        self.plate_type = plate_type
        super(Plate, self).__init__(material)
```

```

    def __repr__(self):
        repr = "Plate\n"
        repr += "Thickness: {}".format(self.thickness)
        repr += "Height: {}".format(self.height)
        repr += "Width: {}".format(self.width)
        repr += "Type: {}".format(self.plate_type)
        return repr

    def min_height_check(self,depth_of_beam):
        """
        Reference: Handbook
        on
        Structural
        Steel
        Detailing, INSDAG - Chapter
        5, Section
        5.2.3, Page 5.7
        """
        if self.height >= 0.6 * depth_of_beam:
            return True
        else:
            return False

    def max_plate_height_check(self,db,tbf,rb1,gap,notch_height,Db,Tbf,Rb1,connectivity):
        """
        Args:
            db - Depth of supported beam
            tbf - Thickness of supported beam flange
            rb1 - Root radius of supported beam flange
            gap - Clearance between fin plate and supported beam flange
            notch_height - max(Tbf, tbf) + max(Rb1, rb1) + max(Tbf / 2, tbf / 2, 10)
            Db - Depth of supporting beam
            Tbf - Thickness of supporting beam flange
            Rb1 - Root radius of supporting beam flange
            connectivity - 'beam-column','beam-beam with single notch' or 'beam-beam with double notch'
        Returns:
            True if plate height >= max_plate_height else False
        """
        notch_height = max(Tbf, tbf) + max(Rb1, rb1) + max(Tbf / 2, tbf / 2, 10)
        if connectivity == 'beam-column':
            max_plate_height = db - 2*(tbf + rb1 + gap)
        if connectivity == 'beam-beam with single notch':
            max_plate_height = db-tbf+rb1 - notch_height
        if connectivity == 'beam-beam with double notch':
            max_plate_height = min(Db, db) - 2 * notch_height
        if self.height <= max_plate_height:
            return True
        else:
            return False

    def min_plate_width_check(self,bf):
        if self.width >= bf:
            return True
        else:
            return False

    def max_plate_width_check(self,bf):
        if self.height <= bf + 25:
            return True
        else:
            return False
    def min_thickness_check(self,F,fy,hp,tw):
        """
        Args:
            F - factored shear force
            fy - yield stress
            hp - height of plate
            tw - thickness of secondary beam web
        Note:
            [Reference: N. Subramanians Design of Steel Structures - Chapter 5, Sec. 5.7.7 - Page 373]
        """
        if self.thickness >= max(tw, 5*F/(fy*hp)):
            return True
        else:
            return False
    def max_thickness_check(self,bolt_diameter):
        """
        Args:
            bolt_diameter
        Returns:
            tp - maximum plate thickness
        Note:
            [Reference: Handbook on Structural Steel Detailing, INSDAG - Chapter 5, Section 5.2.3, Page 5.7]
        """
        if self.thickness <= 0.5 * bolt_diameter:
            return True
        else:
            return False

class Angle(Component):

    def __init__(self, designation, material=Material()):
        self.designation = designation

```

```
        self.leg_a_length = 0.0
        self.leg_b_length = 0.0
        self.thickness = 0.0
        self.Iz = 0.0
        self.Iy = 0.0
        self.cz = 0
        self.cy = 0
        self.root_radius = 0.0
        self.toe_radius = 0.0


        self.connect_to_database_update_other_attributes(designation)

        self.length = 0.0
        super(Angle, self).__init__(material)

    def __repr__(self):
        repr = "Angle\n"
        repr += "Designation: {}".format(self.designation)
        return repr

    def connect_to_database_update_other_attributes(self, designation):
        conn = sqlite3.connect(self.path_to_database)
        db_query = "SELECT AXB, t, R1, R2, Iz, Iy, cz, cy FROM Angles WHERE Designation = ?"
        cur = conn.cursor()
        cur.execute(db_query, (designation,))
        row = cur.fetchone()

        axb = row[0]
        axb = axb.lower()
        self.leg_a_length = float(axb.split("x")[0])
        self.leg_b_length = float(axb.split("x")[1])
        self.thickness = row[1]
        self.root_radius = row[2]
        self.toe_radius = row[3]
        self.cz = row[4]
        self.cy = row[5]
        self.Iz = row[6]
        self.Iy = row[7]


        conn.close()
```

Github Link: Click Here

2.3 Moment of inertia module

A python module is created to calculate the moment of inertia of built up sections with more accuracy. The module would be helpful in calculating the area moment of inertia of tapered built-up I sections with fillet at the corners. In the module the area moment of inertia of each of common shapes and functions of theorems like parallel axis, rotation, translation etc.are added,which are called accordingly to find the moment of inertia of a given shape about some axis.

Chapter 3

Software Testing

The task was to test each and every module with its sub-connectivities currently available in Osdag. There was a need to test the software for edge and corner cases and make sure that the software gave appropriate results/suggestions. Testing of 2-D drawings and design report was equally important. Last but not the least testing of the Gui and other small features was required to ensure smooth functioning of the software. The found bugs were documented and reported to the Osdag team where the members worked on fixing the bugs/issues simultaneously.

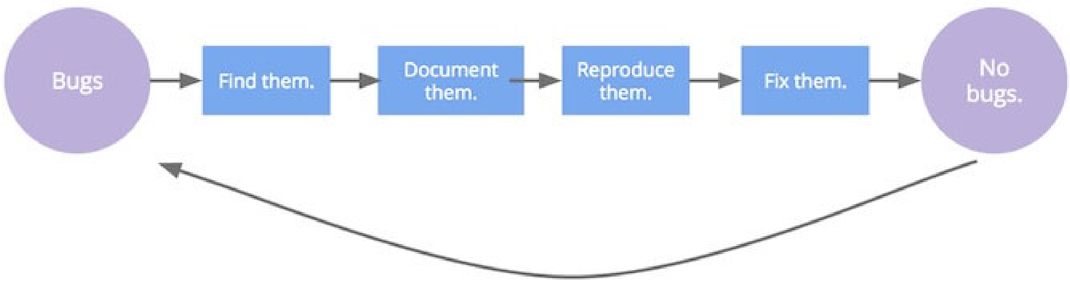
3.1 Creating Inputs

One part was to create the safe design inputs for all the possible design features and bolt combinations. The objective of the task is to test all the features of the features of Osdag including design reports,GUI, 2D models, Cads etc..

3.2 Crash Inputs

One of the task was to find the all type of inputs in which the software is crashing. This helped to locate the bug which is causing the crash. Finding new bugs was really a challenging task, to report any bug statement it was necessary to cross check that statement for every section, connectivity and for various input values. Generally my task follows the following flow chart.

Test the software → Find bugs (If any) → Report to osdag team → Bug fixed by osdag team → Test the software after bug fixed.



All the crash inputs are saved as .osi files so that osdag team can assign and comment over it.

Chapter 4

Conclusion

On the whole, this internship was a useful experience. I have gained new knowledge, skills and met many new people. I achieved several learning goals, and have moved a step further in achieving other. I got insight into professional practice. Internship has proved to be satisfactory and it has allowed as an opportunity to get an exposure of the practical implementation of theoretical fundamentals.

Here during the internship period I developed my skills in following softwares/tools :

1. Osdag
2. Python
3. Object oriented Programming
4. Latex
5. Git and Git hub

Throughout the internship, I have learnt some important skills and qualities like time management, teamwork etc.

I would like to once again appreciate everyone who has made my internship training a superb experience.