Fossee Summer Fellowship Report

On

Osdag

Submitted by

Nikhil Kapoor

Under the guidance of

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

July 23, 2019

Acknowledgment

I would like to thank the FOSSEE project at IIT Bombay for giving me an opportunity to intern for Osdag. This internship was a great opportunity for me to learn and develop myself professionally. It helped me to enhance my knowledge in structural steel design and software development.I am grateful to everyone who helped me during this period.

I would like to especially acknowledge Prof.Siddhartha Ghosh, for being large heart-ed and open minded and who in spite of being busy with his duties, took time out to guide and motivate us and allowed me to work at the esteemed Structural Safety Risk & Reliability(SSRR) Lab. I got a glimpse of Prof.Ghosh's magnanimous leadership, by observing how he maintains a healthy working environment and treats his subordinates with such compassion. I was delighted to see him treating the entire team to delicious cake whenever there was any junior's birthday!

I would also like to thank the members of the entire Osdag team for creating such an amiable working milieu. A special thanks to my mentors, Danish Ansari (Project Research Assistant) and Ajmal Babu MS (Project Research Engineer) and Saurubh Das (Project Research Associate) who patiently helped me with my doubts in steel design and interpreting the IS codes every time I approached him.

Finally I would like to thank everyone at SSRR lab and my fellow interns for making my experience fun and memorable.

Contents

Chapter 1

Introduction to FOSSEE Summer Fellowship (Osdag)

FOSSEE summer fellowship is provided under the FOSSEE project. FOSSEE project promotes the use of FOSS (Free and Open Source Software) to improve quality of education in our country. FOSSEE encourages the use of FOSS tools through various activities to ensure open source alternatives to commercial proprietary software

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

FOSSEE Summer Fellowship is held during May-July. Any UG/PG/PhD holder can apply for this internship. Selection is based on screening tasks. Candidates have to chose a certain FOSS under the FOSSEE Project and complete the required tasks.Interns are then made to work on their chosen FOSS during the fellowship.

Osdag is one such open source software,being developed under the FOSSEE Project at IIT Bombay. Osdag is my FOSS of choice

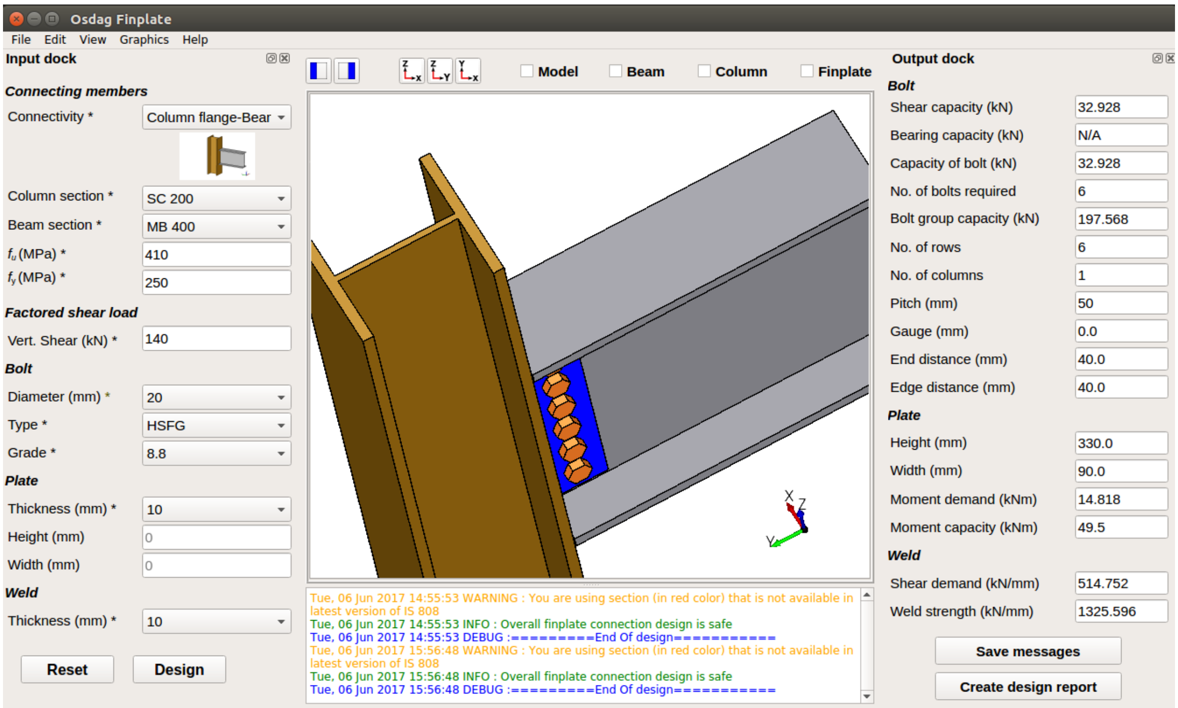
1.1 What is Osdag ?

Osdag is a cross-platform free and open-source software for the design of steel structures, based on the Indian standard IS 800:2007 and developed in Python. It allows the users to design connections, members and systems using a graphical user interface. The interactive GUI provides a 3D visualization of the designed component and creates images for construction/-fabrication drawings.

It is used for solving steel structure problems and to see how the connection will look after practical implementation.

Osdag provides various features such as:

- An interactive window displaying a 3D CAD model, which provides a clear visualization 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.
- 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.

Link to download Osdag:https://osdag.fossee.in/resources/downloads

1.2 Who can use it?

Osdag is being developed for educational and professional use. As Osdag is funded by MHRD for the purpose of education, the main objective is to help students learn steel design. However, Osdag is intended to be developed to the extent that even professionals choose to use it in their regular work.

Osdag is so user friendly and easy to use, that even a novice can start using it. There are video tutorials to get started. These can be accessed at https://osdag.fossee.in/resources/videos.

1.3 Current Stage of Development, Refactoring and Future Plans

Osdag is currently in the nascent stage, although modules on Shear and Moment connection have been developed and are ready to use. The final completed Osdag aims to have modules for:
1.Connection(Shear,Moment and Truss connections)
2.Tension Member
3.Compression Member
4.Flexural Member
5.Beam-Column
6.Plate Girder
7.Truss
8.2D Frame
9.3D Frame
10.Group Design

Apart from development of new modules, the Osdag team is working on Refactoring (restructuring existing computer code without changing its external behaviour) based on the principles of OOP (object oriented programming) so as to improve readability and maintainability of existing code.

Initially the code was written in Python 2 and support for Python 2 ends in 2020. So a separate team of interns worked on transitioning from Python 2 to Python 3.This team of interns also worked on developing the user interface and backend.

1.4 Fellowship Tasks

The following tasks have been completed by me during the Fellowship. These are the topics of the upcoming chapters:
1.Creation of functions for IS800:2007
2.Developing of classes and objects in components file
3.Development of module to find Section Properties of common sections

Chapter 2

Version Control:Git and Github

When multiple people are working on the same files, it is standard professional practice to use version control to allow ease of collaboration, to keep track of changes, to make fixes and add features. The Osdag team uses git and github for version control and collaboration.

As part of our tasks my fellow interns and I had to familiarize ourselves with git and github, so that we can collaborate with rest of the Osdag team. I completed this course on git and github by Udacity:https://www.udacity.com/course/how-to-use-git-and-github–ud775

Given below are links to the main Osdag repositories as well as my personal Osdag repository in which my contributions can be viewed.

1.Main Repository:https://github.com/osdag-admin/Osdag
2.My fork(copy) of the Osdag repository:https://github.com /Nikhil008/Osdag

Chapter 3

Object Oriented Programming Paradigm

Osdag is being developed based on principles of Object Oriented Programming(OOP). OOP is a programming language model in which programs are organized around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior(aka method). In context of Osdag, an example of an object can be a Bolt with attributes such as material, length, diameter grade, type, shear capacity, tension capacity, bearing capacity,etc and behaviours to compute shear capacity, tension capacity, bearing capacity,etc. This opposes the historical approach to programming where emphasis was placed on how the logic was written rather than how to define the data within the logic.

The first step in OOP is to identify all of the objects a programmer wants to manipulate and how they relate to each other, an exercise often known as data modeling. Once an object is known, it is generalized as a class of objects that gives a template of the kind of data that the object can contain and logic sequences to manipulate that data. Each distinct logic sequence is known as a method.

In my case I had to deal with objects such as 'bolts', 'plate','weld',etc, which were used in the component file(Chapter 5).

This approach to programming is well-suited to collaborative development of large complex programs, such as in the case of Osdag. The main benefits of OOP are reusability, scalability and efficiency.

Chapter 4

Creation of functions for IS800:2007

Osdag takes inputs from the user and produces designs based on IS800:2007 code for design of steel construction. Where IS800, does not provide guidelines, other sources such as INSDAG and euro codes are referred. Hence for further development of Osdag, a ready to use set of functions based on the IS code is highly valuable.

We as a team (Tanmmay Kala(IIT Bombay), Rachna Gupta(NIT Silchar) and me) were given the task of developing code for Sections 3,6,7,8 and 9. A part of this task was completed as part of our screening task, wherein we had to write python functions for any one section in IS800. The section chosen by me as part of the screening task was **section 6: Design of Tension Members**.

During the internship in particular I coded entire **section 9: Members Subjected to Combined Forces**, and made several major corrections to **section 7: Design of Compression Members** and **section 8: Design of Members Subjected to Bending**.

The entire code incorporating clauses of IS800 are written as methods of IS800_2007 class within the is800_2007.py file.

4.1 Screening Task Submission

The python code of **section 6: Design of Tension Members** submitted by me for the screening task is as follows

```
#Module for Indian Standard, section 6, IS 800 : 2007
#@author: Nikhil Kapoor id = 161010015


import math

class IS800_2007(object):

    #Perform calculations on steel design as per IS 800:2007

    # ==========================================================
    #          Section 6    Design of Tension Members
    # ==========================================================


    #cl.6.1 Check for factored design tension
    @staticmethod
    def cl_6_1(T,Tdg,Tdn,Tdb):
        #To check if factored design tension is less than design
         ↪  strength
        #Args:
        #    T - factored design tension
        #    Tdg - design strength due to yeilding of gross section
        #    Tdn - design rupture strength of critical section
        #    Tdb - design strength due to block shear
        #Returns:
        #    'OK' if T < Td where Td is minimum of Tdg,Tdn,Tdb
        #    else a warning
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.1

        Td = min(Tdg,Tdn,Tdb)

        if (T<Td): return 'OK'

        else: return 'warning:factored design tension is equal to or
         ↪  exceeds design strength'
```

```
    #cl.6.2 Design strength (yeilding)
    @staticmethod
    def cl_6_2(fy,Ag,gamma_m0=1.1):
        #Calculates design strength due to yielding of gross
         ↪ section(tension)
        #Args:
        #    fy - yield stress of material(N/mm^2)
        #    Ag - gross area of cross-section(mm^2)
        #    gamma_m0 - partial safety factor for failure in tension
         ↪ by yielding
        #Returns:
        #    Tdg -  design strength due to yielding of gross
         ↪ section(N)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.2

        Tdg = Ag * fy / gamma_m0

        return Tdg

    #cl.6.3.1 Design strength due to rupture of critical section -
     ↪ Plates
    @staticmethod
    def cl_6_3_1(fu,b,t,dh,n,g,ps,gamma_m1=1.25):
        #Calculates design strength due to rupture of critical section
         ↪ in plates
        #Args:
        #   gamma_m1 - partial safety factor for failure at ultimate
         ↪ stress
        #   fu - ultimate stress of material(N/mm^2)
        #   b - width of plate(mm)
        #   t - thickness of plate(mm)
        #   dh - diameter of bolt hole in mm (2mm in addition in case
         ↪ of directly
        #        punched holes)
        #   g - a list of gauge lengths between bolt holes(mm)
        #   ps - a list of staggered pitch length between line of bolt
         ↪ holes(mm)
        #   n - number of bolt holes in critical section
        #Returns:
        #    Tdn -  design strength due to rupture of critical section
         ↪ in plates(N)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.3.1

        An = b - n * dh
```

```

        for i in range(n-1):
            An = An + ps[i]**2 / (4*g[i])


        An = An * t  #net effective area of member

        Tdn = 0.9 * An * fu / gamma_m1

        return Tdn

    #cl.6.3.2 Design strength due to rupture of critical section - Threaded Rods
    @staticmethod
    def cl_6_3_2(An,fu,gamma_m1=1.25):
        #Calculates design strength due to rupture of critical section in
        #Threaded Rods
        #Args:
        #    gamma_m1 - partial safety factor for failure at ultimate stress
        #    fu - ultimate stress of material(N/mm^2)
        #    An - net root area of threaded section(mm^2)
        #Returns:
        #     Tdn -  design strength due to rupture of critical section in
        #               threaded rods(N)
        #Note:
        #     Reference:
        #         IS 800:2007, cl.6.3.2

        Tdn = 0.9 * An * fu / gamma_m1

        return Tdn

    #cl.6.3.3 Design strength due to rupture of critical section - Single Angles
    #          (exact formula)
    @staticmethod
    def cl_6_3_3_exact(Anc,Ago,fu,fy,w,bs,Lc,t,gamma_m1=1.25,gamma_m0=1.1):
        #Calculates design ruputre strength at critical section of angle connected
        #through one leg
        #Args:
        #    gamma_m1 - partial safety factor for failure at ultimate stress
        #    gamma_m0 - partial safety factor for failure in tension by yielding
```

```
        #   fu - ultimate stress of material(N/mm^2)
        #   fy - yield stress of material(N/mm^2)
        #   Anc - net area of connected leg(mm^2)
        #   Ago - gross area of outstanding leg(mm^2)
        #   w - width of outstanding leg(mm)
        #   bs - shear lag width(mm)
        #   Lc - length of the end conncetion(mm)
        #   t - thickness of the leg(mm)
        #Returns:
        #    Tdn -  design strength due to rupture of critical section in N (single angle)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.3.3


        beta = 1.4 - 0.076 * (w/t) * (fy/fu) * (bs/Lc)

        if(beta<0.7 or beta>fu*gamma_m0/fy/gamma_m1):
            return 'warning: beta is out of valid range'

        Tdn = 0.9 * Anc * fu / gamma_m1 + beta * Ago * fy / gamma_m0

        return Tdn

    #cl.6.3.3 Design strength due to rupture of critical section - Single Angles
    #          (approximation)
    @staticmethod
    def cl_6_3_3_approx(An,fu,number_of_bolts,gamma_m1=1.25):
        #Calculates design strength due to rupture of critical section in
        #Single Angle for priliminary sizing
        #Args:
        #   gamma_m1 - partial safety factor for failure at ultimate stress
        #   fu - ultimate stress of material(N/mm^2)
        #   An - net area of total cross-section(mm^2)
        #   number_of_bolts - to determine value of alpha (used in calculation)
        #Returns:
        #    Tdn -  approximate design strength due to rupture of critical section in
        #           single angle for preliminary calculation or in absence of detailing(N)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.3.3

```

```python

        if number_of_bolts <=2 :
            alpha = 0.6
        elif number_of_bolts == 3 :
            alpha = 0.7
        elif number_of_bolts >= 4 :
            alpha = 0.8

        Tdn = alpha * An * fu / gamma_m1

        return Tdn

    #cl.6.3.4 Design strength due to rupture of critical section - Other Section
    @staticmethod
    def cl_6_3_4(Anc,Ago,fu,fy,w,bs,Lc,t,gamma_m1=1.25,gamma_m0=1.1):
        #Calculates design ruputre strength at critical section for double angles,
        #channels,I-sections and other rolled steel sections
        #Args:
        #   gamma_m1 - partial safety factor for failure at ultimate stress
        #   gamma_m0 - partial safety factor for failure in tension by yielding
        #   fu - ultimate stress of material(N/mm^2)
        #   fy - yield stress of material(N/mm^2)
        #   Anc - net area of connected leg (mm^2)
        #   Ago - gross area of outstanding leg (mm^2)
        #   w - width of outstanding leg(mm)
        #   bs - shear lag distance,i.e,distance from farthest edge of outstanding leg
        #        to nearest hole/weld line in the connected leg of the cross-section (mm)
        #   Lc - length of the end conncetion(mm)
        #   t - thickness of the leg(mm)
        #Returns:
        #    Tdn -  design strength due to rupture of critical section for double angles,
        #               channels,I-sections and other rolled steel sections(N)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.3.4


         beta = 1.4 - 0.076 * (w/t) * (fy/fu) * (bs/Lc)

         if(beta<0.7 or beta>fu*gamma_m0/fy/gamma_m1):
             return 'warning: beta is out of valid range'
```

```
        Tdn = 0.9 * Anc * fu / gamma_m1 + beta * Ago * fy / gamma_m0

        return Tdn

    #cl.6.4.1 Design strength due to block shear - bolted connections
    @staticmethod
    def cl_6_4_1(Avg,Avn,Atn,Atg,fy,fu,gamma_m0=1.1,gamma_m1=1.25):
        #Calculates design strength due to block shear in bolted connections
        #Args:
        #   gamma_m1 - partial safety factor for failure at ultimate stress
        #   gamma_m0 - partial safety factor for failure in tension by yielding
        #   fu - ultimate stress of material(N/mm^2)
        #   fy - yield stress of material(N/mm^2)
        #   Avg - minimum gross area in shear along bolt line parallel to external force(mm^2)
        #   Avn - minimum net area in shear along bolt line parallel to external force(mm^2)
        #   Atg - minimum gross area in tension from the bolt hole to the toe of the angle,
        #           end bolt line,perpendicular to the line of force(mm^2)
        #   Atn - minimum net area in tension from the bolt hole to the toe of the angle,
        #           end bolt line,perpendicular to the line of force(mm^2)
        #Returns:
        #    Tdn -  design strength due to block shear in bolted connections(N)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.3.4


        Tdb1 = (Avg*fy/math.sqrt(3)/gamma_m0 + 0.9*Atn*fu/gamma_m1)

        Tdb2 = (0.9*Avn*fu/math.sqrt(3)/gamma_m1 + Atg*fy/gamma_m0)

        Tdb = min(Tdb1,Tdb2)

        return Tdb

    #cl.6.4.2 Design strength due to block shear - welded connections
    @staticmethod
    def cl_6_4_2(Av,At,fy,fu,gamma_m0=1.1,gamma_m1=1.25):
```

```
         #Calculates design strength due to block shear in welded connections
        #Args:
        #   gamma_m1 - partial safety factor for failure at ultimate stress
        #   gamma_m0 - partial safety factor for failure in tension by yielding
        #   fu - ultimate stress of material(N/mm^2)
        #   fy - yield stress of material(N/mm^2)
        #   Av - minimum area in shear along bolt line parallel to external force(mm^2)
        #   At - minimum area in tension from the bolt hole to the toe of the angle,
        #        end bolt line,perpendicular to the line of force(mm^2)
        #Returns:
        #    Tdn -  design strength due to block shear in welded connections(N)
        #Note:
        #    Reference:
        #        IS 800:2007, cl.6.3.4

        Tdb1 = (Av*fy/math.sqrt(3)/gamma_m0 + 0.9*At*fu/gamma_m1)

        Tdb2 = (0.9*Av*fu/math.sqrt(3)/gamma_m1 + At*fy/gamma_m0)

        Tdb = min(Tdb1,Tdb2)

        return Tdb
```

4.2 IS800 Python File

The following is a link to the is800_2007 file coded by the team(Tanmay, Rachna and Nikhil). My work and contribution can be scrutinized from the commits of this repository.

https://github.com/Nikhil008/Osdag/blob/refactoring/app/utils/common/is800_2007.py

Chapter 5

Development of classes and objects in components file

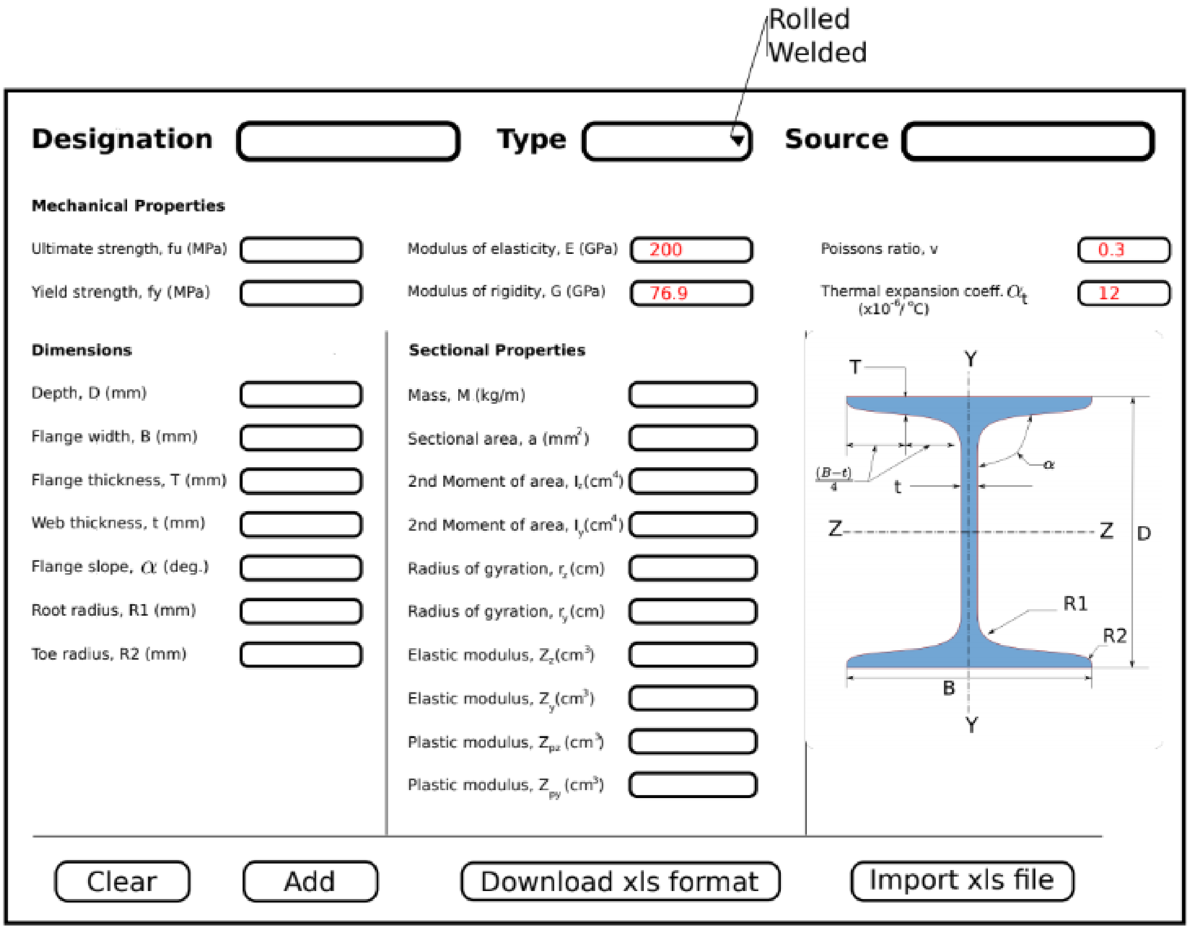
As explained earlier, Osdag is to be structured in the OOP paradigm. The components.py file is intended to be used for calculations pertaining to components such as 'bolt', 'weld', 'nut', 'section', etc. Tanmay Kalla and I were given the task of data modelling for this file by identifying the various Objects (such as 'Bolt','Bolt Group','Weld','Nut',etc) and deciding appropriate attributes and methods to be assigned to them. In turn to code these classes, attributes and methods while referring to functions from the is800 file wherever possible.The following is a github link to the components file coded by us.

https://github.com/Nikhil008/Osdag/blob/components
/app/utils/common/component.py

Chapter 6

Development of a Section Properties module

Section properties such as area moment of inertia, radius of gyration and section modulus(elastic and plastic) are required in calculations for design of steel structures. For standard sections these can be determined by looking up IS808 or relevant steel tables. However if we want to use a section which is not listed in these standards or tables or which may be built-up, we will need to compute these properties for ourselves. In particular I was given the task to find the section properties of a general I-section, from values inputed by the user as shown below.



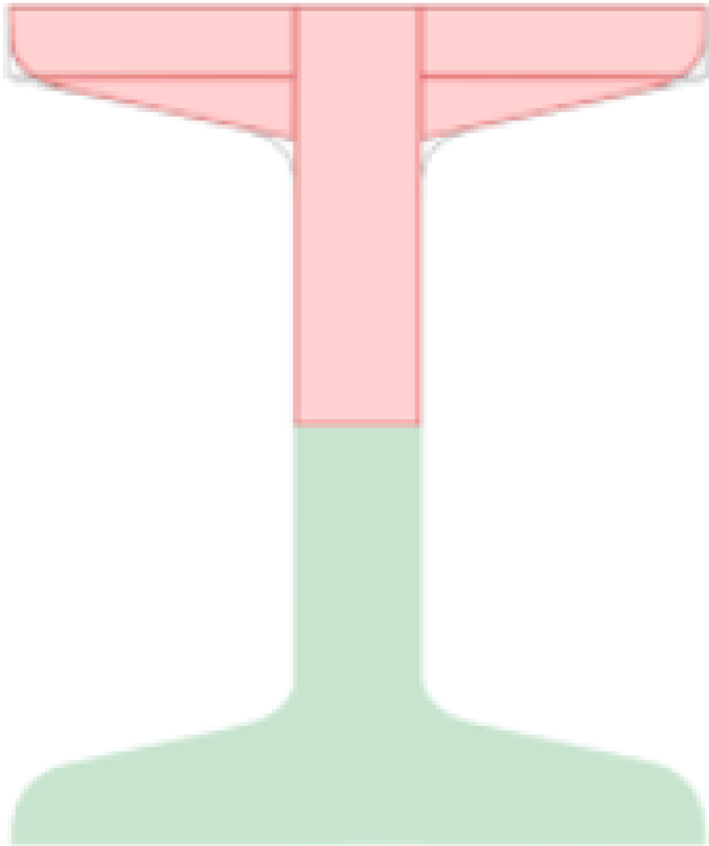
For this purpose I devised the idea of creating a python module which I envision to be used to calculate the section properties of sections of various shapes with general parameters.This will enable us to compute section properties of sections irrespective of whether they are listed or not listed in the standard steel tables or codes.

I have partly developed the aforementioned module, which I shall refer to as *section_properties* to calculate section properties of various shapes as well as the general I-section shown above. Work on plastic section modulus of I-section is yet to be completed.

I would like to point out that their is a lot of scope for expansion. One can build upon this module and develop code for other section shapes such as T-section,C-section, unsymmetrical I-section,etc.

6.1 Methodology and Approach

There are no straight-forward formulae for calculating the section properties of the I-section shown above, as well as many of the other commonly used sections. The approach I used in solving this problem is of division. I divided the problem into a set of simpler sub-problems.
The I-section can be divided into simpler shapes as shown below:



1.Rectangle of type 1 x 1 (web)
2.Rectangle of type 2 x 2 (flange)
3.Triangle x 4 (between flange and web)
4.Fillet of type 1 x 4 (on web)
5.Fillet of type 2 x 4 (on flange)

note: here fillet refers to the shape bounded by an arc(of root or toe radius) and two of its tangents.

All the shapes from 1 to 4 add up and shape 5 is removed to produce the I-section above.

The main challenge here is to compute the area moment of inertia of this section about the X and Y axes. Other properties can be derived from thereof. I created classes for the the following shapes.

1.Shape(abstract class)
2.Rectangle
3.Triangle
4.Sector
5.Fillet
6.I_Section

The following attributes and methods have been defined for each of the classes. The methods basically calculate the properties of the section and assigns it to the corresponding attribute.

$Attributes$:

1.$Centroid$
2.$Area$
3.I_x
4.I_y
5.R_x
6.R_y
7.Ze_x
8.Ze_y
9.Zp_x
10.Zp_y

$Methods$:
1.$centroid()$
2.$area()$
3.$i_x()$

4.$i_y()$
5.$r_x()$
6.$r_y()$
7.$ze_x()$
8.$ze_y()$
9.$zp_x()$
10.$zp_y()$
The following are parameters taken by the constructors of these classes to define the dimensions of the section.

1.Rectangle: length, breadth
2.Triangle: base, height, angle
3.Sector: radius, angle
4.Fillet: radius, angle
5.$I\_Section$: D, B, T, t, R1, R2, alpha

The following classes have been given additional attributes to represent the dimensions of the corresponding section, as well to represent the smaller component shapes that make up the section.

1.Rectangle: Length, Breadth
2.Triangle: Base, Height, Angle
3.Sector: Radius, Angle
4.Fillet: Radius, Angle, FilletSector, FilletTriangle
5.$I\_Section$: D, B, T, t, R1, R2, alpha, $Fillet_1, Fillet_2,$
$Rectangle_1, Rectangle_2, Triangle_1$

The explanation of the methods of these classes are given in the sections to follow.

6.2 Functions for Centroid Calculation, Parallel Axis Theorem and Rotation Transformation of Area Moment of Inertia

In the *section_properties* module I have created global functions for calculation of centroids of composite shapes, to implement parallel axis theorem and rotational transformation of area moment of inertia

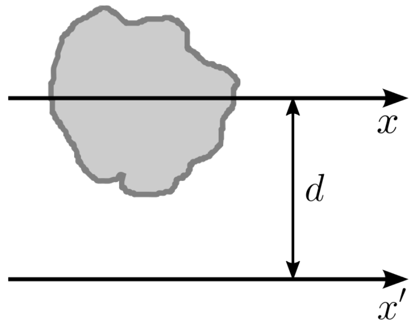
Centroid Calculation

The coordinates (x or y) of the centroid of a composite of shapes can easily be attained by the following formula:

$$X = \sum A_i x_i / \sum A_i$$

where,
the A's are the areas of the composing shapes and
the x's are the coordinates of the centroid of the composing shape

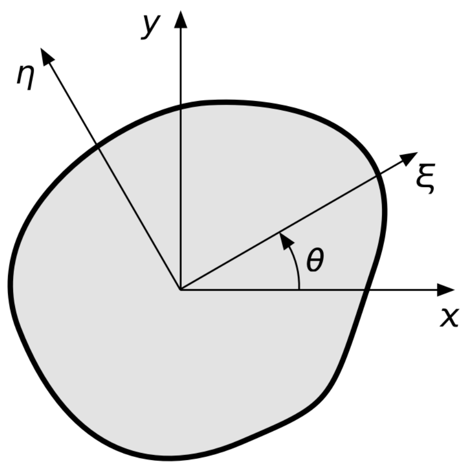
Parallel Axis Theorem



The area moment of inertia I of a shape of area A about an axis x', parallel to centroidal axis, x and a distance d away, with Icm being the area moment of inertia about x, is given by

$$I = Icm + Ad^2$$

Rotational Transform of area moment of inertia



The formula for finding the area moment of inertia about coordinate axes, ξ and η at angle θ anticlockwise from orthogonal axes x and y is

$$I_{\xi\xi} = \frac{I_{xx} + I_{yy}}{2} - \frac{I_{yy} - I_{xx}}{2} \cos 2\theta - I_{xy} \sin 2\theta$$

$$I_{\eta\eta} = \frac{I_{xx} + I_{yy}}{2} + \frac{I_{yy} - I_{xx}}{2} \cos 2\theta + I_{xy} \sin 2\theta$$

where, $I_{\xi\xi}, I_{\eta\eta}, I_{xx}$ and I_{yy} are the area moment of inertia about ξ, η, x and y respectively, and I_{xy} is the product of inertia about x and y axes.

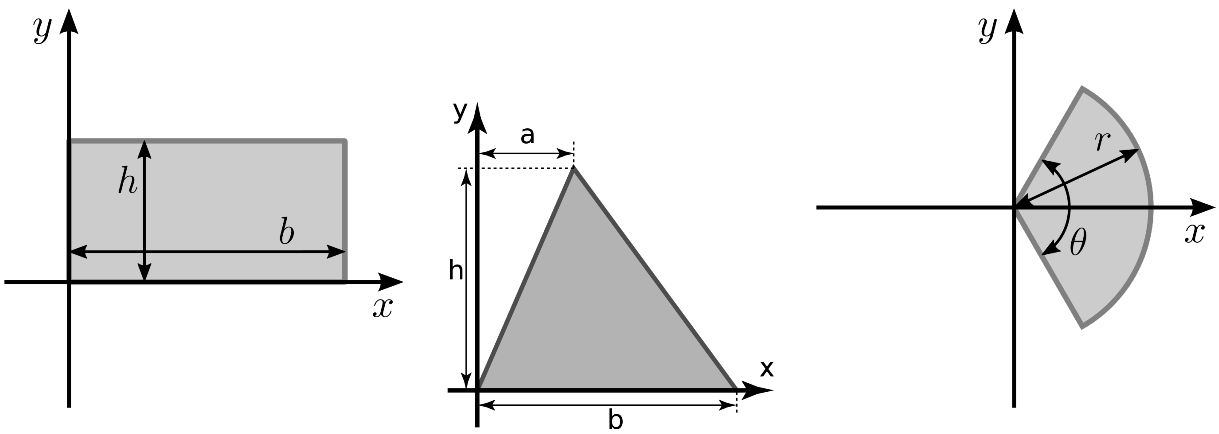
A snippet from the *section_properties* module showing the implementation of the theorems/equations above:

```
import math
import numpy as np
import abc


def parallel_axis_transform_i(i,area, d):
    return i + area * d**2


def rotational_transform_i(i_x, i_y, i_xy, phi):
    i_u = (i_x + i_y)/2 + (i_x - i_y)/2 * math.cos(2*phi) - i_xy *
     ↪  math.sin(2*phi)
    i_v = (i_x + i_y) / 2 - (i_x - i_y) / 2 * math.cos(2 * phi) + i_xy
     ↪  * math.sin(2 * phi)
    return i_u, i_v


def calc_centroid(areas, coordinates):
    """
    :param areas: list of areas of composing shapes
    :param coordinates: list of either x or y coordinates of
↪   centroids of respective shapes
    :return: x/y coordinate of  centroid of composite shape
    """
```

```python
    areas = np.array([areas])
    coordinates = np.array([coordinates])
    return sum(areas * coordinates)
```

6.3 Rectangle, Triangle, Sector

The common attributes and corresponding functions to assign values to the attributes for Rectangle, Triangle and Sector are listed below. The attributes for each shape are defined with respect to the coordinate diagrams shown below.Refer below for the meaning of the attributes(which is the same as the return value of the corresponding function).



note: In the code, in case of the triangle, instead of parameter 'a' I have used 'alpha', where 'alpha' is the angle between the 2 lines of the triangle that meet at the origin.

1.Centroid, centroid()
    tuple representing coordinates of centroid
2.Area, area()
    area of shape/section
3.I_x, i_x()

area moment of inertia about axis parallel to x axis and passing through centroid

4.I_y, i_y()

area moment of inertia about axis parallel to y axis and passing through centroid

5.R_x, r_x()

radius of gyration about axis parallel to x axis and passing through centroid

6.R_y, r_y()

radius of gyration about axis parallel to y axis and passing through centroid

7.Ze_x, ze_x()

elastic section modulus about axis parallel to x axis

8.Ze_y, ze_y()

elastic section modulus about axis parallel to y axis

9.Zp_x, zp_x()

plastic section modulus about axis parallel to x axis

10.Zp_y, zp_y()

plastic section modulus about axis parallel to y axis

The content of these functions in the Rectangle, Triangle and Fillet class are based on straight forward formulae one can find on the links below, and can be understood easily.

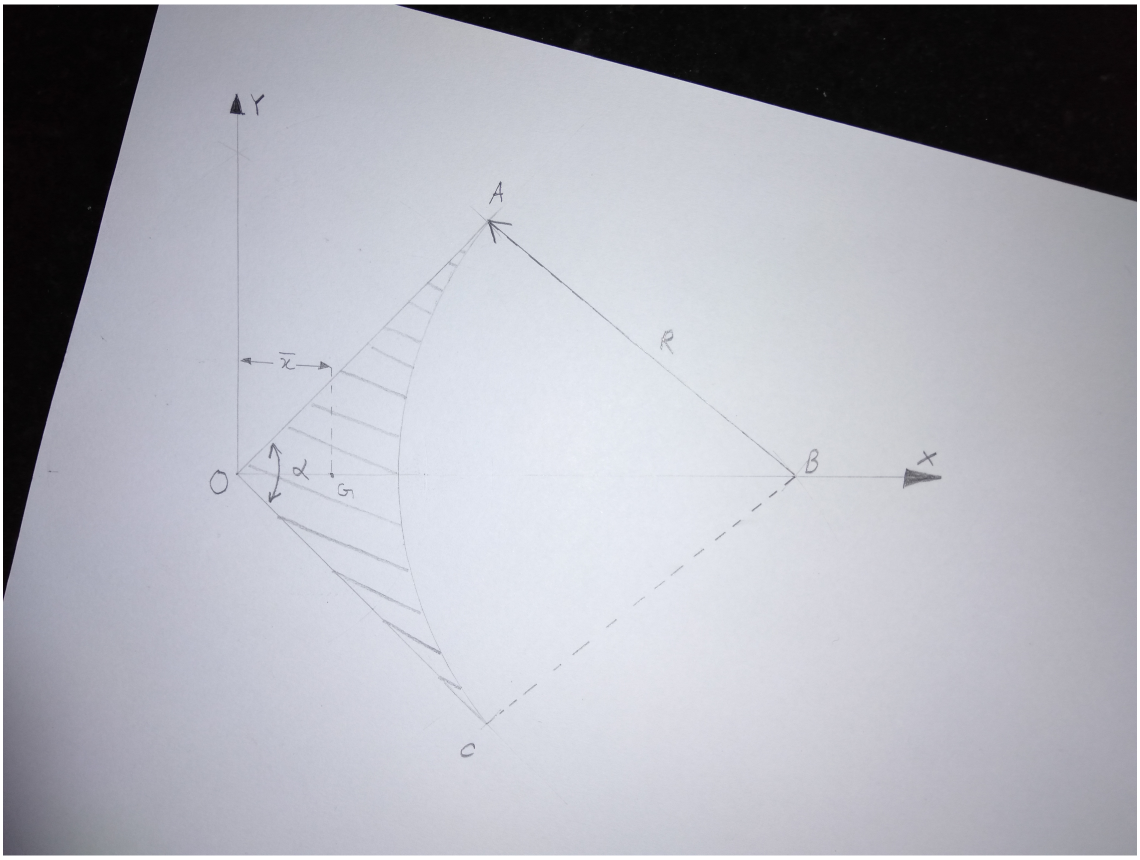
list of centroids
list of second-moments of area
list of section-moduli

In the following sections I explain how I develop functions to calculate the properties of Fillet and I-section as the process of computation in these cases is more involved.

6.4 Fillet

I must first clarify what I refer to as a fillet over here. For lack of a better word, I define fillet as the region bounded by two intersecting lines and a circular arc tangent to both these lines.

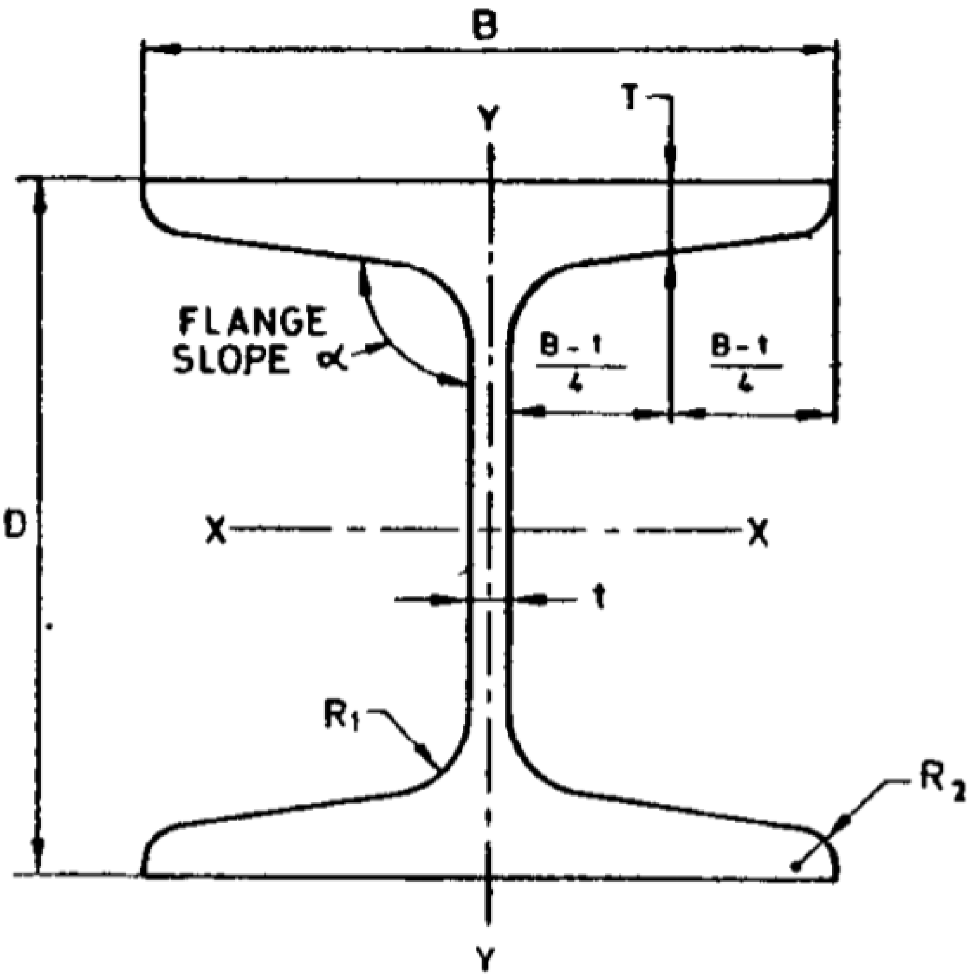
The definition of the attributes in the Fillet class, are same as that of Rectangle,Triangle and Sector classes, except for the coordinate system and parameters as shown below:



I have considered the fillet to be composed of 2 traingles in addition and a sector in subtraction. For this I created the attributes, FilletSector and FilletTriangle, which are essentially instances of the Triangle and Sector classes respectively.

6.5 I-Section

The parameters taken by an I-section object and the x and y axes considered are depicted as:



The attributes of the I-section hold the same meaning as the previously mentioned classes.

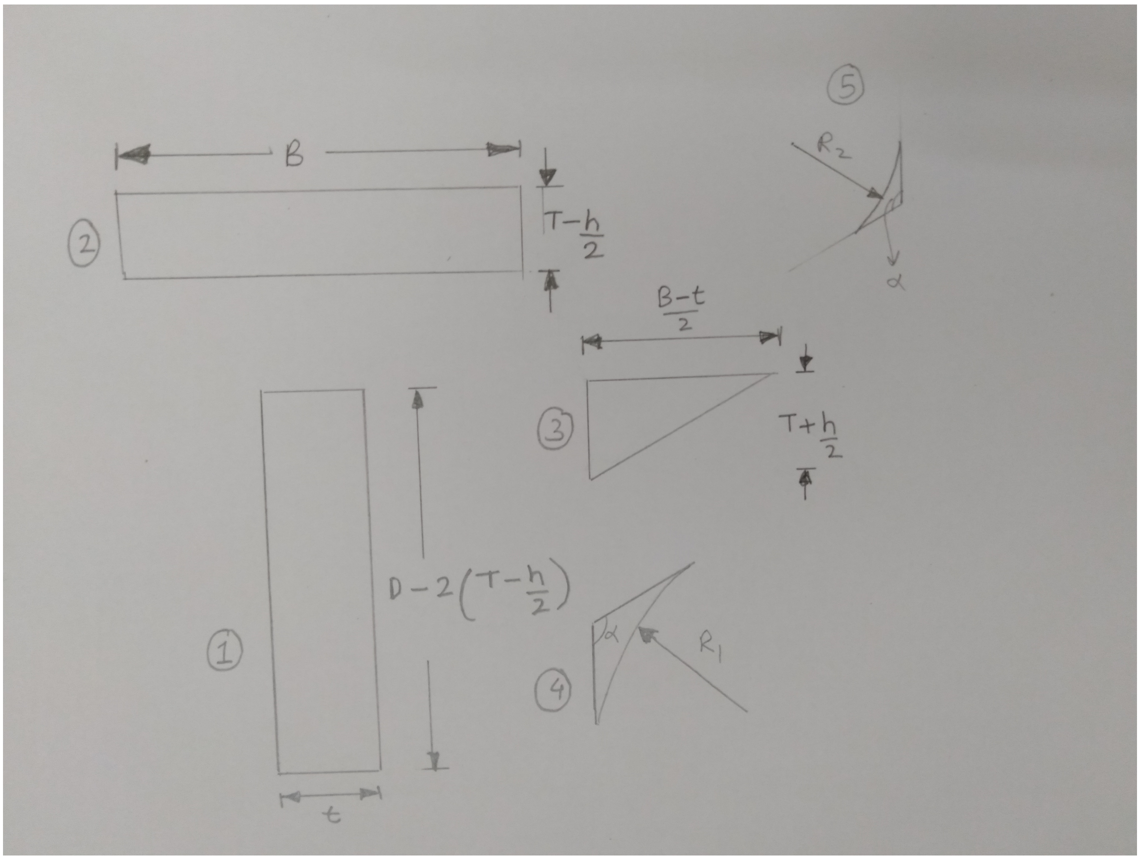
Rectangle (Type 1) refers to the rectangle which forms the web. Rectangle (Type 2) refers to the rectangles which form the flanges.The 4 Triangles are in the space between the web and the flanges.Fillets (Type 1) are those which are formed between the Triangle and Web,while Fillets (Type 2), are those between the Triangle and the flanges.

Calculation for attributes like area and centroid(origin itself) are self-explanatory.The main challenge is the calculation of the area moments of inertia.From this the radius of gyration and elastic section modulus can easily be calculate.(work on plastic section modulus needs to be completed).So, given below is explanation for the calculation of area moment of inertia of the I-section.

Area Moment of Inertia Calculation Steps

The I-section is composed of the following shapes as explained earlier:
1.Rectangle of type 1 x 1
2.Rectangle of type 2 x 2
3.Triangle x 4
4.Fillet of type 1 x 4
5.Fillet of type 2 x 4

From these the contribution for the first 4 shapes need to be summed up and that of the 5th shape needs to be subtracted.The dimensions of these shapes in terms of parameters of I-section class and $h = \frac{(B-t)}{2}\cot(\pi - \alpha)$ are depicted below.



As explained earlier, apart from the attributes of the I-section that represent dimension we have additional attributes $Fillet\_1$, $Fillet\_2$, $Rectangle\_1$, $Rectangle\_2$ and $Triangle\_1$, which are instances of the Fillet, Rectangle and Triangle classes.

Let X and Y refer to the main coordinate axes of the I section and x and y to the the coordinate axes of the composing shapes about their centroids(as explained earlier).

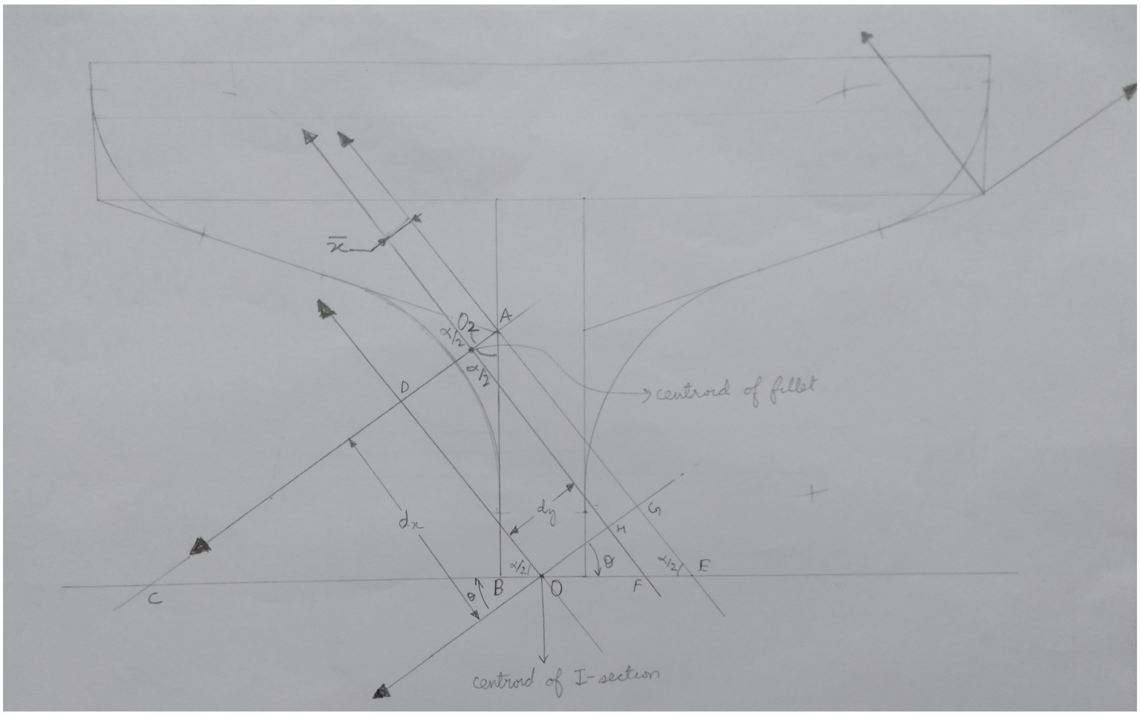
First we find the area moment of inertia of each of the shapes about the X and Y axes and then sum up the contributions of each of the shapes about the X-Y axes.

The x and y axes for the first 3 shapes are parallel to that of the X and Y axes of the I-section and the moments of inertia of theses shapes can be easily expressed about the X and Y axes by application of parallel axis theorem.

In case of the Fillet shapes we first apply the parallel axis theorem to find an expression for the moment of inertia of these shapes about parallel axes about the centroid of the I-section and then we apply rotational transform to get the moment of inertia contribution about X and Y axes.This needs some explanation:

$Fillet_1$

To apply parallel axis theorem we need d_x and d_y.We can consider we already have $\overline{x}$, as we have the instance $Fillet\_1$.Also, we can express AB and BO in terms of our Rectangle and Triangle instances.So essentially we need d_x and d_y given AB,BO and $\overline{x}$.



By geometry,

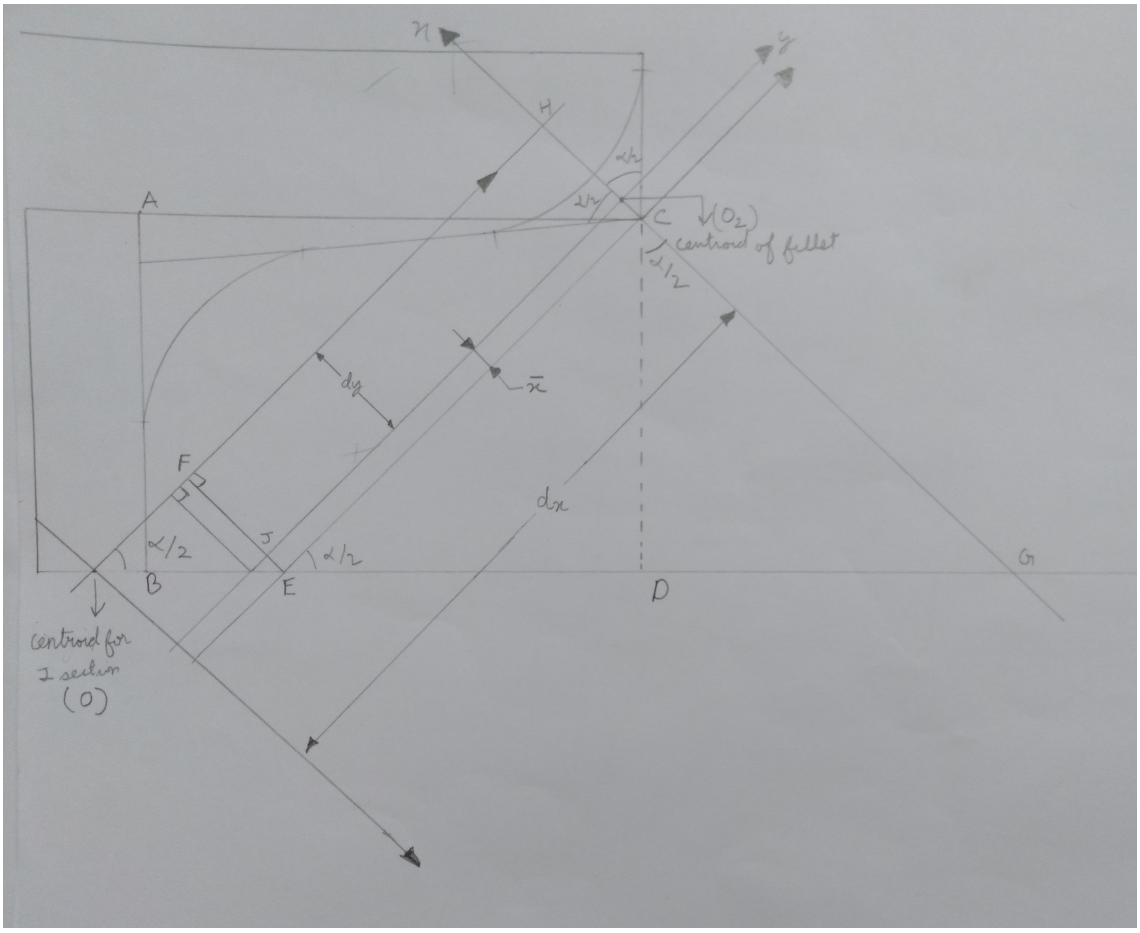
$$\begin{gathered}CB = AB\tan(\alpha/2)\\ CO = CB + BO\\ DO = CO\cos(\alpha/2)\\ d_x = DO\end{gathered}$$

Similarly,

$$\begin{gathered}BE = AB\cot(\alpha/2)\\ OE = BE - BO\\ GO = OE\sin(\alpha/2)\\ DA = GO\\ DO_2 = DA - \overline{x}\\ d_y = DO_2\end{gathered}$$

After applying parallel axis Theorem we apply a rotation transform of $\theta = \pi/2 - \alpha/2$. The product of inertia of fillet is zero from symmetry.

$Fillet_2$



Again to apply parallel axis theorem we need d_x and d_y.We can consider we already have $\overline{x}$,AB,OB and AC which we can express in terms of the Fillet, Rectangle and Triangle instances.

By geometry,

$$CD = AB$$
$$GD = CD\tan(\alpha/2)$$

$$DB = AC$$
$$GO = GD + DB + OB$$
$$OH = GO\cos(\alpha/2)$$
$$d_x = OH$$

Similarly,

$$ED = CD\cot(\alpha/2)$$
$$OE = OB + DB - ED$$
$$FE = OE\sin(\alpha/2)$$
$$FJ = FE - \overline{x}$$
$$d_y = FJ$$

After applying parallel axis Theorem we apply a rotation transform of $\theta = \pi/2 - \alpha/2$. The product of inertia of fillet is zero from symmetry.

Refer to the complete section-properties module implemented by me at https://github.com/Nikhil008/Section_properties/blob/master/section_properties.py

Chapter 7

References

1. IS 800:2007
2. IS 808
3. Design of Steel Structures N. Subramanian
4. https://calcresource.com/moment-of-inertia-rotation.html
5. *https://en.wikipedia.org/wiki/List_of_centroids*
6. *https://en.wikipedia.org/wiki/List_of_second_moments_of_area*
7. *https://en.wikipedia.org/wiki/Section_modulus*

Chapter 8

Conclusion

This fellowship was a highly rewarding experience. I acquired new knowledge and skills and got connected to new people, who guided me and made my experience pleasurable.

I got an insight into professional practice in software development. I am fortunate to have had the opportunity to implement my knowledge gained from my civil engineering courses to the development of Osdag.