Summer Fellowship Report

On

**Python code for different sections of IS 800:2007
and
Preparing DDCL for Truss connection**

Submitted by

Rachana Gupta

Under the guidance of

Prof.Siddhartha Ghosh
Civil Engineering Department
IIT Bombay

Under the Mentorship of

Sourabh Das
Project Research Associate

July 13, 2019

Acknowledgment

I would like to thank FOSSEE for providing me a platform to work on something I am very interested in. I am thankful to everyone who thought of having and involved in selection process based on screening tasks. I am grateful to be a part of team which promotes open source software.

I thank all the Osdag members, who are wonderful mentors and great team. I thank Sourabh Das (Project Research Associate), Ajmal Babu MS (Project Research Associate), Danish Ansari (Project Research Assistant), Yash Lokhande (Project Research Assistant), Darshan Viswakarma (Project Research Associate), Anand Swaroop (Project Research Associate), Anjali Jatav (Project Research Assistant) Deepti Reddy(Co-intern) and whole team, who made us feel welcome and planned all the tasks meticulously during this period.

I am grateful that I got a chance to work under Prof. Sidharth Ghosh, who took time to mentor us and monitored individual contributions as well.

Contents

Chapter 1

Introduction

1.1 FOSSEE Summer Research Fellowship

FOSSEE project promotes the use of FOSS (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 availability of competent free software equivalent to commercial (paid) softwares.
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. Osdag internship is provided through FOSSEE project. Any UG/PG/PhD holder can apply for this internship. And the selection will be based on a screening task.

1.2 What is Osdag?

Osdag is Free/Libre and Open Source Software being developed for design of steel structures. Its source code is written in Python, 3D CAD images are developed using PythonOCC. Github is used to ensure smooth workflow between different modules and team members. It is in a path where people from around the world would be able to contribute to its development. FOSSEE's "Share alike" policy would improve the standard of the software when the source code is further modified based on the industrial and educational needs across the country.
Design and Detailing Checklist (DDCL) for different connections, mem-

bers and structure designs is one of the important bi-products of this project. It would create a repository and design guide book for steel construction based on Indian Standard codes and best industry practices.

1.3 Who can use ?

Osdag is created both for educational purpose and industry professionals. As Osdag is currently funded by MHRD, Osdag team is developing 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.

Osdag can be used by anyone starting from novice to professionals. It's simple user interface makes it flexible and attractive than other software. Video tutorials are available to help get started. The video tutorials of Osdag can be accessed here.

Chapter 2

Python Code of IS 800: 2007

We are creating library for fuctions of IS 800: 2007, so that it can be used directly for compiling code of various modules of Osdag in future. So, I converted sections of IS 800: 2007 to python codes.

For creating python codes google strings are used to explain variables and output,where 'Args' section is used for defining variables and 'Return' is for output for each functions of IS 800:2007, as shown in figure;

```python
# cl 7.1.2.1 design compressive stress of axially loaded member
@staticmethod
def cl_7_1_2_1_design_compressive_stress(K_L,alpha,E,f_y,r,gamma_m0):
    """
        Calculation of design compressive stress
    Args:
            K_L  - Effective length of compression member in mm
            alpha - Imperfection factor
            E - Young's Modulus of Elasticity in N/mm**2
            f_y - Yield Stress in N/mm**2
            r - radius of gyration in mm

        Return:
            f_cd - Design  strength of compression member in N/mm**2

        Note:
            Reference:
            IS 800:2007, cl.7.1.2.1
    """

    f_cc = (math.pi ** 2 * E) / (K_L / r) ** 2  #euler buckling stress
    lambda_ = math.sqrt(f_y / f_cc)  #non-dimensional slenderness ratio
    phi = 0.5 * (1 + alpha * (lambda_ - 0.2) + lambda_ **2)
    srf = 1 / (phi + math.sqrt(phi ** 2 - lambda_ ** 2))  #stress reduction factor,kai
    f_cd = min( f_y / gamma_m0 * srf, f_y / gamma_m0)
    return f_cd
```

Tables of each sections of IS 800:2007 are stored as dictionary in python code, as shown below;

```python
# Table 5 Partial Safety Factors for Materials, gamma_m (dict)
class IS800_2007(object):
    cl_5_4_1_Table_5 = {"gamma_m0": {'yielding': 1.10, 'buckling': 1.10},
                        "gamma_m1": {'ultimate_stress': 1.25},
                        "gamma_mf": {'shop': 1.25, 'field': 1.25},
                        "gamma_mb": {'shop': 1.25, 'field': 1.25},
                        "gamma_mr": {'shop': 1.25, 'field': 1.25},
                        "gamma_mw": {'shop': 1.25, 'field': 1.50}
                        }
```

2.1 Python Code for Section 6 IS 800: 2007

Tension members are linear members in which axial forces acts and cause elongation, how such members should be designed is given in Section 6 ('Design of Tension Members') of IS 800: 2007 and its functions are converted into python codes so that it can be directly used in future for the development of module for design of tension Member in Osdag.It's Python codes is attached vide Appendix - A.

2.2 Python Code for Section 7 IS 800: 2007

Compression members are the members which can take axial compressive load. These members usually fail by flexure buckling and section 7 of IS 800: 2007 explains how such members should be designed to resist buckling failure. Python code for this is attached vide Appendix - B

2.3 Python Code for Section 8 IS 800: 2007

Members subjected to predominant bending must have adequate design strength to resist bending moment and shear force and how such member should be designed properly so that it can work efficiently in its design period is explained in section 8 ("Design of Members subjected to bending") of IS 800: 2007. Python code for this is attached vide Appendix - C

2.4 Python Code for Annex E IS 800: 2007

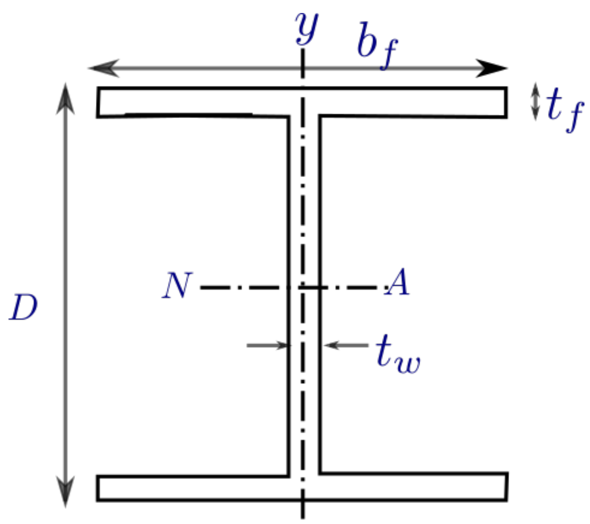
Factors affecting elastic critical moment and it's calculation constitutes Annex E ("Elastic Lateral Torsional Buckling") of IS 800: 2007.
Python code for this is attached in vide Appendix - D
Git link for this is attached here: https://github.com/Rachnagupta29/Osdag/pull/1

Chapter 3

Sectional Properties of I- section

3.1 Creating class for sectional properties of I section

Class of sectional properties for I section is created so that Osdag software can also be used for the design of sections which are not in Steel Tables. And users will have freedom to design section according to their preference.



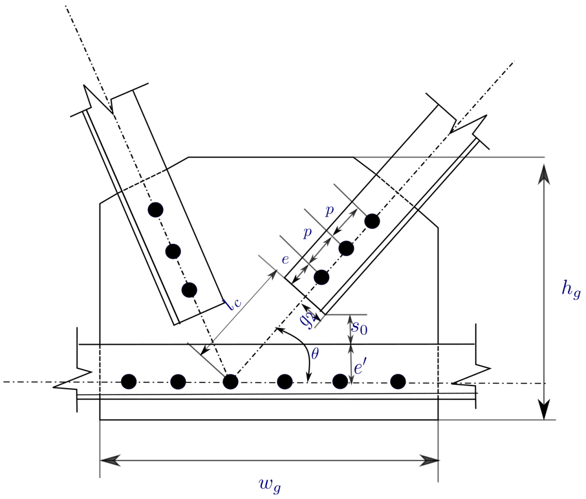
Python code for this is attached vide Appendix - E

Chapter 4

Preparation of DDCL

4.1 DDCL for Bolted Truss connection

I have made DDCL for Bolted Truss connection, which is going to be the design module of Osdag software in future. By this designers will be able to design efficient Bolted connection for truss for various geometrical arrangements.



The DDCL of Truss connection is attached vide Appendix - F

4.2 DDCL for Welded Truss connection

I prepared DDCL for weld connection, in which gusset plate is connected to members through welding in truss connection.

The DDCL of Truss connection is attached vide Appendix -G

Chapter 5

Reference

- IS 800:2007 Indian Standard Code of Practice for General Construction in Steel, 2007.
- IS 808:1989 Indian Standard Code of Practice for Dimensions or Hot Rolled Steel Beam, Column, Channel and Angle Sections,1989.
- US Department of Transportation Federal Highway Administration, Guidelines for Design and Rating of Gusset-Plate Connections for Steel Truss Bridges.
- William A. et al, The Whitmore Section.
- Institute of Steel Development and Growth. Hand Book on Structural Steel Detailing, 2018a.
- N.Subramanian Design of Steel Structures. Oxford University Press, 13 edition, 2013.
- American Institute of Steel Construction. Steel Construction Manual, 13 edition, 2017.

Chapter 6

Conclusion

The internship enriched me with knowledge, skills and gave me a chance to interact with new people, which helped me to achieved several learning goals, and have moved a step further in achieving other. I got insight into how design is done in filed and the adverse factors that need to be considered. Internship has proved to be satisfactory and it has allowed as an opportunity to get an exposure of the practical implementation of theoretical fundamentals.

I learned technical things, like Osdag, Basics of Python, Latex, Git and Git hub

Throughout the internship, I found that several things are important:

- **Critical and Analytical Thinking**

  To organize our tasks and assignment, we need to analyse our problems and assignment, and to formulate a good solution to the problem. We would have to set contingency plan for the solution, so that we are well prepared for the unforeseeable situations.

- **Time Management**

  As overall project staff and programmer are always racing against tight time line and packed schedule, a proper time management will minimize facing overdue deadlines. An effective time management allows us to do our assignment efficiently and meet our schedules. Scheduling avoids time wastage and allows us to plan ahead, and gaining more as a result.

- **Goal Management**

It is better to sub-divide the goals to a few achievable tasks, so that we will be gaining more confidence by accomplishing those tasks.

- **Colleague Interactions**

  In working environment, teamwork plays a vital role in contributing to a strong organization. Teamwork is also essential in reaching the goals of the organization as an entity. Thus, communicating and sharing is much needed in the working environment. Therefore, we should be respecting each other in work, and working together as a team, instead of working alone. This is because working together as a team is easier in reaching our targets, rather than operating individually.

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

Appendices

Appendix A

Python Code for Section 6 IS: 800 2007

```
import math
"""    SECTION  6     DESIGN OF TENSION MEMBERS   """
# Table 5 Partial Safety Factors for Materials, gamma_m (dict)
class IS800_2007(object):
    cl_5_4_1_Table_5 = {"gamma_m0": {'yielding': 1.10, 'buckling': 1.10},
                        "gamma_m1": {'ultimate_stress': 1.25},
                        "gamma_mf": {'shop': 1.25, 'field': 1.25},
                        "gamma_mb": {'shop': 1.25, 'field': 1.25},
                        "gamma_mr": {'shop': 1.25, 'field': 1.25},
                        "gamma_mw": {'shop': 1.25, 'field': 1.50}
                        }


# DESIGN OF TENSION MEMBER
# cl 6.1 Tension
@staticmethod
def cl_6_1_Design_strength_of_tesion_member(T_dg, T_dn, T_db):
    """
        Calculation of Design Strength of the member as per cl.6.1
    Args:
        T_dg - design strength due to yielding of gross section in N (float),
        T_dn - design rupture strength of critical section in N (float),
        T_db - design strength due to block shear in N (float)

    Return:
        T_d -   design strength of the number under axial tension in N (float),
    Note:
        References:
        IS800:2007,cl.6.1
    """
    T_d = min(T_dg, T_dn, T_db)
    return T_d


#cl.6.2 Design Strength due to yielding of Gross Section
@staticmethod
def cl_6_2_Design_strength_of_member_due_to_yielding_of_gross_section(A_g, f_y):
    """
        Calculate the design strength due to yielding of gross section as
```

```
    per cl.6.2
    Args:
        A_g- Gross area of cross section[in sq.mm](float)
        f_y- Yield stress of the material in Mpa (float)

    Return:
        T_dg - Design strength of member due to yielding of gross section
    in N(float)


    Note:
        Reference:
        IS 800:2007, cl.6.2
    """

    gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
    T_dg = A_g * f_y / gamma_m0
    return T_dg


# cl.6.3 Design Strength Due to Rupture of Critical Section
# cl.6.3.1 Plates
@staticmethod
def cl_6_3_1_design_strength_in_tension(b, n, d_h, p_s, g, f_u, t):
    """
        Calculate the design strength in tension of a plate as per cl.6.3.1
    Args:
        A_n- net effective area of the member[in sq.mm](float)
        f_u- ultimate stress of the material in Mpa(float)
        b,t- width and thickness of the plate,respectively[in mm](float)
        d_h- diameter of the bolt hole[2mm in addition to the diameter of
    the hole,
                    in case the directly punched holes[in mm](float)
        g-  gauge length between the bolt holes[in mm](float list)
        p_s- staggered pitch length between line of bolt holes[in mm](float
     list)
        n- number of bolt holes in the critical section (int)

    Return:
        T_dn - design strength in tension of a plate in N(float)

    Note:
        Reference:
        IS 800:2007, cl.6.3.1
    """

    gamma_m1 = IS800_2007.cl_5_4_1_Table_5["gamma_m1"]['ultimate_stress']
    sum_value = 0
    for i in range(n - 1):
        sum_value += (p_s[i] * p_s[i]) / (4 * g[i])

    A_n = (b - n * d_h + sum_value) * t
    T_dn = 0.9 * A_n * f_u / gamma_m1
    return T_dn


# cl.6.3.2 Threaded Rods
@staticmethod
def cl_6_3_2_design_strength_of_threaded_rods_in_tension(A_n, f_u):
    """
        Calculate the design strength of threaded rods in tension as per cl
```

```
    .6.3.2

    Args:
        A_n- net root area at the threaded section [in sq.mm](float)
        f_u- ultimate stress of the material in Mpa(float)

    Return:
            T_dn - design strength of threaded rods in tension in N(float)

    Note:
        Reference:
        IS800:2007, cl.6.3.2
    """

    gamma_m1 = IS800_2007.cl_5_4_1_Table_5["gamma_m1"]['ultimate_stress']
    T_dn = 0.9 * A_n * f_u / gamma_m1
    return T_dn


# cl.6.3.3 Single Angles
@staticmethod
def cl_6_3_3_design_strength_of_an_angle_connected_through_one_leg(A_nc,
    f_u, w, t, f_y, b_s, L_c, A_go):
    """
        Calculation of design strength of an angle connected through one
    leg as per cl.6.3.3

    Args:
        w -   outstanding leg width in mm(float)
        b_s -  shear lag width in mm(float)
        L_c - length of the end connection, that is the distance between
    the outermost
                bolts in the end joint measured along the load direction or
     length of the weld
                along the load direction in mm(float)
        A_nc - net area of the connected leg[in sq. mm](float)
        A_go - gross area of the outstanding leg[in sq. mm](float)
        t -    thickness of the leg in mm(float)
        f_u-  ultimate strength of material in Mpa(float)

    Return:
        T_dn - design strength of an angle connected through onw leg in N(
    float)

    Note:
        Reference:
        IS800:2007, cl.6.3.3

    """

    gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
    gamma_m1 = IS800_2007.cl_5_4_1_Table_5["gamma_m1"]['ultimate_stress']

    X = max(0.7, f_u * gamma_m0 / f_y * gamma_m1)
    beta = min(X, 1.4 - 0.076 * (w / t) * (f_y / f_u) * (b_s / L_c))
    T_dn = (0.9 * A_nc * f_u / gamma_m1) / (beta * A_go * f_y / gamma_m0)
    return T_dn


# cl 6.3.3 For preliminary sizing
@staticmethod
```

```
def cl_6_3_3_1_design_strength_of_net_section(n, A_n, f_u):
    """
    Calculation of rupture strength of net section for preliminary sizing
    as per cl.6.3.3

    Args:
        A_n - net area of the total cross-section[in sq.mm](float)
        f_u - ultimate tensile strength pf material in Mpa(float)
        n  -  number of bolts (int)

    Return:
        T_dn - design strength of net section for preliminary sizing in N(
    float)

    Note:
        Reference:
        IS800:2007 cl.6.3.3
    """
    if n in [1, 2]:
        alpha = 0.6
    else:
        if n == 3:
            alpha = 0.7
        else:
            alpha = 0.8

    gamma_m1 = IS800_2007.cl_5_4_1_Table_5["gamma_m1"]['ultimate_stress']
    T_dn = alpha * A_n * f_u / gamma_m1
    return T_dn


#cl.6.3.4 Other Section
#cl.6.4.1 Block shear strength of bolted connections
@staticmethod
def cl_6_4_1_block_shear_strength(A_vg, A_vn, A_tg, A_tn, f_u, f_y):
    """
        Calculation of the block shear strength of bolted connection as per
     cl.6.4.1

    Args:
        A_vg - Minimum gross in shear along bolt line parallel to external
    force [in sq.mm](float)
        A_vn - Minimum net area in shear along bolt line parallel to
    external force[in sq.mm](float)
        A_tg - Minimum gross area in tension from the bolt hole to the toe
    of the angle,
                end bolt line, perpendicular to the line of force,
    respectively [ in sq.mm](float)
        A_tn - Minimum net area in tension from the bolt hole to the toe of
     the angle,
                end bolt line, perpendicular to the line of force,
    respectively [in sq.mm]
        f_u -  Ultimate stress of the plate ,material in Mpa(float)
        f_y -  Yield stress of the plate material in Mpa(float)

    Return:
        T_db - block shear strength of bolted connection in N (float)

    Note:
        Reference:
        IS800:2007, cl.6.4.1
```

```
    """
    gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
    gamma_m1 = IS800_2007.cl_5_4_1_Table_5["gamma_m1"]['ultimate_stress']
    T_db1 = A_vg * f_y / (math.sqrt(3) * gamma_m0) + 0.9 * A_tn * f_u /
    gamma_m1

    T_db2 = 0.9 * A_vn * f_u / (math.sqrt(3) * gamma_m1) + A_tg * f_y /
    gamma_m0
    T_db = min(T_db1, T_db2)

    return T_db

# ====================================================
```

Appendix B

Python Code for Section 7 IS: 800 2007

```
"""    SECTION  7     DESIGN OF COMPRESS1ON MEMBERS    """
import math
# Table 5 Partial Safety Factors for Materials, gamma_m (dict)
class IS800_2007(object):
    cl_5_4_1_Table_5 = {"gamma_m0": {'yielding': 1.10, 'buckling': 1.10},
                        "gamma_m1": {'ultimate_stress': 1.25},
                        "gamma_mf": {'shop': 1.25, 'field': 1.25},
                        "gamma_mb": {'shop': 1.25, 'field': 1.25},
                        "gamma_mr": {'shop': 1.25, 'field': 1.25},
                        "gamma_mw": {'shop': 1.25, 'field': 1.50}
                        }


    # cl 7.1 Design Strength
    # cl.7.1.2
    @staticmethod
    def cl_7_1_2_design_copmressive_strength_of_a_member(A_c, f_cd):
        """
            Calculation of design compressive strength
        Args:
            A_c - Effective sectional area (in square mm)
            f_cd - Design Compressive stress(in N)

        Return:
            P_d - Design  Compressive Strength of a member (in N)

        Note:
            References:
            IS800:2007 cl.7.2
        """

        P_d = f_cd * A_c
        return P_d


    # cl 7.1.2.1 design compressive stress of axially loaded member
    @staticmethod
    def cl_7_1_2_1_design_compressive_stress(K_L,alpha,E,f_y,r,gamma_m0):
        """
            Calculation of design compressive stress
        Args:
                K_L  - Effective length of compression member in mm
```

```
                alpha - Imperfection factor
                E - Young's Modulus of Elasticity in N/mm**2
                f_y - Yield Stress in N/mm**2
                r - radius of gyration in mm

            Return:
                f_cd - Design  strength of compression member in N/mm**2

            Note:
                Reference:
                IS 800:2007, cl.7.1.2.1
        """

        f_cc = (math.pi ** 2 * E) / (K_L / r) ** 2  #euler buckling stress
        lambda_ = math.sqrt(f_y / f_cc)  #non-dimensional slenderness ratio
        phi = 0.5 * (1 + alpha * (lambda_ - 0.2) + lambda_ **2)
        srf = 1 / (phi + math.sqrt(phi ** 2 - lambda_ ** 2))  #stress reduction factor,kai
        f_cd = min( f_y / gamma_m0 * srf, f_y / gamma_m0)
        return f_cd


    # cl 7.1.2.2 Calculation of buckling class of given cross-section
    @staticmethod
    def cl_7_1_2_2_Table_10_Buckling_class_of_cross_section(Cross_section, t_f,t_w,h,b_f):
        """
            Defining Buckling Class of Cross-Section
        Args:
            Cross_section - Either 'Rolled_I_Section' or 'Welded_I_Section'
                            or 'Hot_rolled_hollow' or 'cold_Formed_hollow' or 'Welded_Box_Section'
                            or 'Channel,Angle,T,Solid Section' or 'Built_up_Member'

            h- Depth of the section in mm
            b_f - width of flange or width of section in case of welded box section(mm)
            t_f - Thickness of flange in mm
            t_w - Thickness of web in mm

        Return:
            Dictionary of Buckling axis and Buckling class with Buckling axis as key

        Note:
            Reference:
            IS 800:2007, cl.7.1.2.2, Table_10
        """
        if Cross_section == "Rolled_I_Section":
            if h / b_f > 1.2 :
                if t_f <= 40:
                    return {'z-z': 'a','y-y': 'b'}

                if t_f>40 and t_f<=100:
                    return {'z-z': 'b','y-y': 'c'}

            if h / b_f <= 1.2:
                if t_f <= 100:
                    return {'z-z': 'b','y-y': 'c'}

```

```
                if t_f>100:
                    return {'z-z': 'd','y-y': 'd'}

        if Cross_section == "Welded_I_Section":
            if t_f <= 40:
                return {'z-z': 'b','y-y': 'c'}
            if t_f>40:
                return {'z-z': 'c','y-y': 'd'}

        if Cross_section == "Hot_rolled_hollow":
            return {'z-z': 'a','y-y': 'a'}

        if Cross_section == "cold_Formed_hollow":
            return {'z-z': 'b','y-y': 'b'}

        if Cross_section == "Welded_Box_Section":
            Buckling_Class_1 = 'b'
            Buckling_Class_2 = 'b'

            if b_f / t_f < 30:
                Buckling_Class_1 = "c"

            if h / t_w < 30:
                Buckling_Class_2 = "c"

            return {'z-z': Buckling_Class_1,'y-y': Buckling_Class_2}

        if Cross_section == "Channel_Angle_T_Solid_Section" or Cross_section == "Built_up_Member":
            return {'z-z': 'c','y-y': 'c'}


    #Imperfection Factor, alpha
    # alpha for a given buckling class,'a','b','c' or 'd'
    cl_7_1_Table_7_alpha = {
        'a': 0.21,
        'b': 0.34,
        'c': 0.49,
        'd': 0.76,
    }

    #Table 11 Effective Length of Prismatic Compression Members
    @staticmethod
    def cl_7_2_2_table11_effective_length_of_prismatic_compression_members(L,BC=[]):

        """
            Effective length of Prismatic Compression Member when the boundary conditions in the plane of buckling
            can be assessed

        Args:
            BC - linked list of Boundary Conditions
                 =[BC_translation_end1,BC_rotation_end1,BC_translation_end2,BC_rotation_end2]
            L -  Length of the Compression member in mm

        Return:
            K_L - Effective length of Compression Member in mm

        Note:
```

```python
            Reference:
            IS 800:2007, cl.7.2.2, Table_11
        """

        if BC == ['Restrained','Restrained','Free','Free'] or BC == ['Restrained','Free','Free','Restrained']:K_L = 2.0 * L
        elif BC == ['Restrained','Free','Restrained','Free']:K_L = L
        elif BC == ['Restrained','Restrained','Free','Restrained']:K_L = 1.2 * L
        elif BC == ['Restrained','Restrained','Restrained','Free']:K_L = 0.8 * L
        elif BC == ['Restrained','Restrained','Restrained','Restrained']:K_L = 0.65 * L
        return K_L


    # cl 7.1.2.1 design compressive stress of axially loaded member
    @staticmethod
    def design_compressive_stress(f_y, r ):

        """
            Calculation of design compressive stress
            Args:
                K_L  - Effective length of compression member in mm
                alpha - Imperfection factor
                E - Young's Modulus of Elasticity in N/mm**2
                f_y - Yield Stress in N/mm**2
                r - radius of gyration of the section in mm

            Return:
                f_cd - Design  strength of compression member in N/mm**2

            Note:
                Reference:
                IS 800:2007, cl.7.1.2.1
        """
        Buckling_Class = cl_7_1_2_2_Table_10_Buckling_class_of_cross_section(Cross_section, h, b_f, t_f)
        alpha = cl_7_1_Table_7_alpha["Buckling_Class"]["alpha"]
        K_L = cl_7_2_2_effective_length_of_prismatic_Compression_members(At_one_end_Translation, At_one_end_Rotation,
                                                                          At_other_end_Translation,
                                                                          At_other_end_Rotation, L)
        f_cc = (pi * pi * E) / (K_L / r) ** 2
        lambda_c = math.sqrt(f_y / f_cc)
        phi = 0.5 * (1 + (alpha * (lambda_c - 0.2)) + (lambda_c * lamda_c))
        srf = 1 / (phi + math.sqrt(phi ** 2 - lambda_c ** 2))
        f_cd = min(((f_y / gamma_m0) * srf), f_y / gamma_m0)
        return f_cd


    # Design of Column Base
    #cl.7.4.3 thickness of column base
    @staticmethod
    def cl_7_4_3_1_Calculation_of_thickness_of_column_base(w,a,b,t_f,f_y):

        """
            Calculation of thickenss of Column base
```

```
        Args:
            w - uniform pressure from below on the slab base in mm
            a - Larger Projection in mm
            b - Smaller Projection in mm
            f_y - Yield Stress in N/mm**2
            t_f - Thickness of flange of Compression member in mm

        Return:
            t_f - thickness of rectangular slab column base in mm

        Note:
            Reference:
            IS 800:2007  cl.7.4.3.1,


        """
        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yield_stress']
        t_s = max(t_f, math.sqrt(2.5 * w * (a ** 2 - 0.3 * b ** 2) * gamma_m0 / f_y))
        return t_s

    #cl.7.5.1.2 Loaded through one angle
    #Table 12 - evaluation of constants K1,K2,K3 for effective slenderness ratio
    @staticmethod
    def cl_7_5_1_2_table12_constant_K_1_K_2_K_3(No_of_Bolts_at_Each_End_Connection, Connecting_member_Fixity):

        """Value of constant K_1,K_2, K_3
        Args:
            No_of_Bolts_at_Each_End_Connection -  Either more than 2 or 1,
            Fixity - Either Fixed or Hinged.

        Return:
            [K_1,K_2,K_3]

        Note:
            Reference:
            IS 800:2007 cl.7.5.1.2


        """

        if No_of_Bolts_at_Each_End_Connection >= 2:
            if Connecting_member_Fixity == "Fixed":
                K_1 = 0.20
                K_2 = 0.35
                K_3 = 20

            elif Connecting_member_Fixity == "Hinged":
                K_1 = 0.70
                K_2 = 0.60
                K_3 = 5

        if No_of_Bolts_at_Each_End_Connection == 1:
            if Connecting_member_Fixity == "Fixed":
                K_1 = 0.75
                K_2 = 0.35
                K_3 = 20

            if Connecting_member_Fixity == "Hinged":
```

```
                K_1 = 1.25
                K_2 = 0.50
                K_3 = 60

        return [K_1,K_2,K_3]

    #cl.7.5.1.2.Design strength of angle strut loaded through one leg
    @staticmethod
    def cl_7_5_1_2_Calculation_of_design_strength_of_single_angle_strut_loaded_through_one_leg(L, b_1, b_2, f_y, r_vv, t, E,K_list):
        """
            Calculation of design strength of single angle strut loaded through one leg

        Args:
            L - Length of Angle section in mm
            b_1,b_2 - width of legs of angle section in mm
            f_y - yield stress in N/mm**2
            r_vv - radius of gyration about minor axis in mm
            t - thickness of the leg in mm
            E - Young's Modulus of elasticity in N/mm***2
            epsilon  -  yield stress ratio

        Return:
            f_cd - Design compressive strength of the section

        Note:
            Reference:
            IS 800:2007  cl.7.5.1.2

        """
        [K_1,K_2,K_3] = K_list

        alpha = 0.49 #according to ammendment 1

        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        epsilon = math.sqrt(250 / f_y)

        lambda_vv = (L / r_vv) / (epsilon * math.sqrt(math.pi ** 2 * E / 250))

        lambda_phi = (b_1 + b_2) / (2 * t * epsilon * math.sqrt(math.pi * math.pi * E / 250))

        lambda_e = math.sqrt(K_1 + (K_2 * lambda_vv ** 2) + (K_3 * lambda_phi ** 2))

        phi = 0.5 * (1 + alpha * (lambda_e - 0.2) + lambda_e ** 2)
        f_cd = min(f_y / (gamma_m0 * (phi + math.sqrt(phi ** phi - lambda_e ** 2))), f_y / gamma_m0)

        return f_cd

    #cl7.6 Laced column
    #cl 7.6.1.5.Effective slenderness ratiion of lacing member
    @staticmethod
    def effective_slenderness_ratio_of_lacing_member(K_L, r_min):
        """
            Calculation of Effective slenderness ratio of lacing member
            to account for shear deformation
```

```
        Args:
            K_L -effective length of column in mm
            r_min - radius of gyration of column member in mm
            SR_0 - actual maximum slenderness ration of column
        Returns:
            SR_eff - effective slenderness ratio of lacing
        Note:
            Reference:
            IS 800:2007  cl 7.6.1.5
        """

        SR_0 = K_L/r_min
        SR_eff = 1.05*SR_0
        return SR_eff

    #cl 7.6.2 Width of Lacing  Bars
    @staticmethod
    def cl_7_6_2_width_of_lacing_bars(d):
        """
        Calculation of min width of Lacing  Bars
        Args:
            d - nominal bolt/rivet diameter
        Returns:
            w_min - min Width of Lacing  Bars
        Note:
            Reference:
            IS 800:2007, cl 7.6.2
        """

        w_min = 3*d
        return w_min

    #cl 7.6.3 Thickness of Lacing Bars
    @staticmethod
    def cl_7_6_3_minimum_thickness_of_lacing_bars(lacing_type,L_eff):
        """
            Calculation of  min Thickness of Lacing Bars
        Args:
            Lacing_type - either 'single_lacing' or 'double_lacing'
            L_eff - effective length of lacing bars
        Returns:
            t_min - minimum thickness of Lacing Bars
        Note:
            Reference:
            IS 800:2007, cl 7.6.3
        """
        if lacing_type == 'single_lacing':
            t_min = 1/40 *L_eff

        else:
            t_min = 1 / 60 * L_eff

        return t_min

    #Cl7.6.6 Design of lacing
    #7.6.6.1 Transverse shear in the lacing bar
    @staticmethod
    def cl_7_6_6_1_transverse_shear_in_the_lacing_bar(P):
        """
            Calculation of Transverse shear in the lacing bar
        Args:
```

```
            P  -  axial load on column in N
        Returns:
            V_t_min - minimum design transverse shear in N
        Note:
            Reference:
            IS 800:2007 cl 7.6.6.1
        """
        V_t_min = 2.5/100 * P

        return V_t_min


    # cl7.7  Batten plate
    # cl7.7.1.4 effective slenderness ratio of batten plate
    @staticmethod
    def Cl_7_7_1_4_effective_slenderness_ratio_of_batten_plate(K_L,r_min):
        """
        Calculation of effective slenderness ratio
        Args:
            K_L -effective length of column in mm
            r_min - minimum radius of gyration(r_x,r_y,r_z) of column member in mm
        Returns:
            SR_eff - effective slenderness ratio of lacing
        Note:
            Reference:
            IS 800:2007, cl 7.6.1.5

        """
        SR_eff = 1.1 * (K_L / r_min)
        return SR_eff

    # Design of Battens
    # Battens
    @staticmethod
    def cl_7_7_2_1_longitudinal_shear_transverse_shear_and_moment_at_connection(P, S, C, N):
        """
            Calculation of longitudinal shear, transverse shear and moment at connetion
            Args:
                P - total axial force on column in N
                S - minimum transverse distance between the centroid of the rivet/bolt
                        group/welding connecting the batten to the main member in mm
                N  -number of parallel planes of battens
                C  -  distance between centre -to- centre of battens in mm
            Returns:
                V_t - transverse shear force in N
                V_b - longitudinal shear force along column axis in N
                M - moment at connection in N*mm
            Note:
                Reference:
                IS 800:2007, cl.7.7.2.1
        """

        V_t = 2.5 / 100 * P
        V_b = V_t * C / (N * S)
        M = V_t * C / (N * 2)
```

```
        return (V_t, V_b, M)
#------------------------------------------------------------------------
```

Appendix C

Python Code for Section 8 IS: 800 2007

```
#
    ==========================================================================

import math
# Table 5 Partial Safety Factors for Materials, gamma_m (dict)
class IS800_2007(object):
    cl_5_4_1_Table_5 = {"gamma_m0": {'yielding': 1.10, 'buckling': 1.10},
                        "gamma_m1": {'ultimate_stress': 1.25},
                        "gamma_mf": {'shop': 1.25, 'field': 1.25},
                        "gamma_mb": {'shop': 1.25, 'field': 1.25},
                        "gamma_mr": {'shop': 1.25, 'field': 1.25},
                        "gamma_mw": {'shop': 1.25, 'field': 1.50}
                        }

    """    SECTION  8     DESIGN OF MEMBERS SUBJECTED TO BENDING   """
    # -------------------------------------------------------------
    # DESIGN OF MEMBER SUBJECTED TO BENDING

    # cl 8.3.3 Effective length for cantilever Beam
    @staticmethod
    def cl_8_3_3_Table_16_Effective_length_for_cantilever_beam(L,
    Restraint_Condition_1, Restraint_Condition_2,

    Loading_condition):
        """
            Calculate effective length for cantilever beam of projecting
    length L as per cl.8.3.3

            Args:
                L - Projecting Length of cantilever beam in mm (float)

        Args:
            L - Projecting Length of cantilever beam in mm (float)
            D -  Overall depth of he beam in mm (float)

            Restrained_condition - Either "At support" or "At Top"
            Restraint_Condition_1- "At support"
            Restraint_Condition_2- "At Top"

                At_support -  Either "Continuous, with lateral restraint to
     top"
                              or "Continuous, with  partial torsional
```

```
restraint"
                          or "Continuous, with lateral and torsional
restraint "
                          or "Restrained laterally,torsionally and
against rotation on plan "

            At_top - Either  "Free"
                    or "Lateral restraint to top flange"
                    or "Torsional restraint"
                    or  "Lateral and torsional restraint"

            Loading_condition - Either "Normal" or  "Destabilizing"

        Returns:
            L_LT =
cl_8_3_3_Table_16_Effective_length_for_cantilever_beam
        Note:
            References:
            IS800:2007, Table 16 (cl 8.3.3)
    """

    if Restraint_Condition_1 == "Continuous, with lateral restraint to
top flage":
        if Restraint_Condition_2 == "Free":
            if Loading_condition == "Normal":
                return 3.0 * L
            else:
                return 7.5 * L
        if Restraint_Condition_2 == "Lateral restraint to top flage":
            if Loading_condition == "Normal":
                return 2.7 * L
            else:
                return 7.5 * L
        if Restraint_Condition_2 == "Torsional restraint":
            if Loading_condition == "Normal":
                return 2.4 * L
            else:
                return 4.5 * L
        if Restraint_Condition_2 == "Lateral and Torsional restraint":
            if Loading_condition == "Normal":
                return 2.1 * L
            else:
                return 3.6 * L
    if Restraint_Condition_1 == "Continuous,with partial torsional
restraint":
        if Restraint_Condition_2 == "Free":
            if Loading_condition == "Normal":
                return 2.0 * L
            else:
                return 5.0 * L
        if Restraint_Condition_2 == "Lateral restraint to top flange":
            if Loading_condition == "Normal":
                return 1.8 * L
            else:
                return 5.0 * L
        if Restraint_Condition_2 == "Torsional restraint":
            if Loading_condition == "Normal":
                return 1.6 * L
            else:
                return 3.0 * L
        if Restraint_Condition_2 == "Lateral and Torsional restraint":
```

```
                if Loading_condition == "Normal":
                    return 1.4 * L
                else:
                    return 2.4 * L
        if Restraint_Condition_1 == "Continuous,with lateral and torsional restraint":
            if Restraint_Condition_2 == "Free":
                if Loading_condition == "Normal":
                    return 1.0 * L
                else:
                    return 2.5 * L
            if Restraint_Condition_2 == "Lateral restraint to top flage":
                if Loading_condition == "Normal":
                    return 0.9 * L
                else:
                    return 2.5 * L
            if Restraint_Condition_2 == "Torsional restraint":
                if Loading_condition == "Normal":
                    return 0.8 * L
                else:
                    return 1.5 * L
            if Restraint_Condition_2 == "Lateral and Torsional restraint":
                if Loading_condition == "Normal":
                    return 0.7 * L
                else:
                    return 1.2 * L
        if Restraint_Condition_1 == "Restrained laterally,torsionally and against rotation on plan":
            if Restraint_Condition_2 == "Free":
                if Loading_condition == "Normal":
                    return 0.8 * L
                else:
                    return 1.4 * L
            if Restraint_Condition_2 == "Lateral restraint to top flage":
                if Loading_condition == "Normal":
                    return 0.7 * L
                else:
                    return 1.4 * L
            if Restraint_Condition_2 == "Torsional restraint":
                if Loading_condition == "Normal":
                    return 0.6 * L
                else:
                    return 0.6 * L
            if Restraint_Condition_2 == "Lateral and Torsional restraint":
                if Loading_condition == "Normal":
                    return 0.5 * L
                else:
                    return 0.5 * L


    @staticmethod
    def cl_8_3_1_Table_15_Effective_length_for_simply_supported_beams(L,D, Restraint_Condition_1,
                                                                      Restraint_Condition_2, Loading_Condition):
        """
            Calculate effective length against lateral torsional buckling for simply supported Beams and girders
            where no lateral restraint  to the compression flange is provided as per cl.8.3.1

```

```
        Args:
            L -  Span of simply supported beams and girders in mm (float)
            D -  Overall depth of he beam in mm (float)

            Restraint_Condition - Either "Torsional Restraint" or "warping Restraint"
            Restraint_Condition_1- "Torsional Restraint"
            Restraint_Condition_2- "Warping Restraint"

            "Torsional Restrained" - Either "Fully restrained" or
                                    "Partially restrained by bottom flange support connection" or
                                    "Partially restrained by bottom flange bearing support"

            "Warping_Restraint" - Either "Both flanges fully restrained" or
                                 "Both flanges partially restrained" or
                                 "Compression flange fully restrained" or
                                 "Compression flange partially restrained" or
                                 "Warping not restrained in both flange"


            Loading_Condition  - Either "Normal" or " Destabilizing"
            Returns:
            L_LT  - cl_8_3_1_Effective length for simply supported Beams in mm (float)
        Note:
                References:
                IS800:2007, Table 15 (cl 8.3.1)
    """

    if Restraint_Condition_1 == "Fully Restrained":
        if Restraint_Condition_2 == "Both flanges partially restrained":
            if Loading_Condition == "Normal":
                return 0.70 * L
            else:
                return 0.85 * L
        if Restraint_Condition_2 == "Compression flange fully Restrained":
            if Loading_Condition == "Normal":
                return 0.75 * L
            else:
                return 0.90 * L
        if Restraint_Condition_2 == "Both flanges fully restrained":
            if Loading_Condition == "Normal":
                return 0.80 * L
            else:
                return 0.95 * L
        if Restraint_Condition_2 == "Compression flange partially Restrained":
            if Loading_Condition == "Normal":
                return 0.85 * L
            else:
                return 1.00 * L

```

```
            if Restraint_Condition_2 == "Warping not restrained in both flanges":
                if Loading_Condition == "Normal":
                    return 1.00 * L
                else:
                    return 1.20 * L
        if Restraint_Condition_1 == "Partially restrained by bottom flange support connection":
            if Restraint_Condition_2 == "Warping not restrained in both flanges":
                if Loading_Condition == "Normal":
                    return 1.00 * L + 2 * D
                else:
                    return 1.20 * L + 2 * D
        if Restraint_Condition_1 == "Partially restrained by bottom flange bearing support":
            if Restraint_Condition_2 == "Warping not restrained in both flanges":
                if Loading_Condition == "Normal":
                    return 1.2 * L + 2 * D
                else:
                    return 1.4 * L + 2 * D


    @staticmethod
    def cl_8_3_Effective_length_against_torsional_restraint(L, D, Beam_type, Restraint_Condition_1,
                                                            Restraint_Condition_2,
                                                            Loading_Condition):
        """
            Calculation of effective length for given type of beam type as per cl.8.3

        Args:

            L-  Span of simply supported beams and girders in mm (float) for
                "Simply_supported_with_no_lateral_restrained_to_the_compression_flanges",
                 Projecting Length of cantilever beam in mm (float) for
                 "Cantilever_beam",
                 Length of relevant segment between the lateral restraint in mm (float) for
                 "Simply_supported_with_intermediate_lateral_restraints",
                 Centre-to-centre distance of the restraint member in mm (float) for
                 'Beam provided with members to give effective lateral restraint to compression flange at interval'

            Beam_type - Either "Simply_supported_with_no_lateral_restrained_to_the_compression_flanges"
                        or "Simply_supported_with_intermediate_lateral_restraints"
                        or "Beam_provided_with_members_to_give_effective_lateral_restrain_to_compression_flange_at_
"
                        or "Cantilever_beam"

```

```
            FOR "
    Simply_supported_with_no_lateral_restrained_to_the_compression_flanges"
            Restraint_Condition - Either "Torsional Restraint" or "warping
    Restraint"

            Restraint_Condition - Either "Torsional Restraint" or "Warping
    Restraint"
            Restraint_Condition_1- "Torsional Restraint"
            Restraint_Condition_2- "Warping Restraint"


            "Torsional Restrained" - Either "Fully restrained" or
                                            "Partially restrained by bottom
    flange support connection" or
                                            "Partially restrained by bottom
    flange bearing support"

            "Warping_Restraint" - Either "Both flange fully restrained" or
                                         "Compression flange fully restrained"
    or
                                         "Compression flange partially
    restrained" or
                                         "Warping not restrained in both flange
    "


            FOR "Cantilever_beam":

            Restrained_condition - Either "At support" or "At Top" for "
    Cantilever_beam"

            Restraint_Condition_1- "At support"
            Restraint_Condition_2- "At Top"

            At_support -  Either "Continuous, with lateral restraint to top
    "
                                 or "Continuous, with  partial torsional
    restraint"
                                 or "Continuous, with lateral and torsional
    restraint "
                                 or "Restrained laterally,torsionally and
    against rotation on plan "

            At_top - Either "free"
                            or "lateral restraint to top flange"
                            or "Torsional restraint"
                            or  "Lateral and torsional restraint"

            Loading_condition - Either "Normal" or  "Destabilizing"



        Returns :
            L_LT - effective_length_of_beam in mm (float)

        Note:
            References:
            IS800:2007,  cl 8.3.

        """

```

```
        if Beam_type == "Simply_supported_with_no_lateral_restrained_to_the_compression_flanges":
            L_LT = IS800_2007.cl_8_3_1_Table_15_Effective_length_for_simply_supported_beams(L, D, Restraint_Condition_1,
                                                                                              Restraint_Condition_2,
                                                                                              Loading_Condition)
        elif Beam_type == 'Simply supported with intermediate lateral restraints':
            L_LT = 1.2 * L

        elif Beam_type == "Beam_provided_with_members_to_give_effective_lateral_restrain_to_compression_flange_at_
":
            L_LT = 1.2 * L #TODO:doubt-check
        else:
            L_LT = IS800_2007.cl_8_3_3_Table_16_Efective_length_for_cantilever_beam(L, Restraint_Condition_1,
                                                                                     Restraint_Condition_2, Loading_Condition)

        return L_LT

    # Design Strenth in Bending(Flexure)
    @staticmethod
    def cl_8_2_Design_strength_in_bending(M, M_d):
        """ Calculation of design bending strength
        Args:
             M: Factored design moment in N*mm
            M_d: design bending strength of the section in N*mm
        Return:
            M_d - design bending strength of the section in N*mm
        Note:
            References:
                IS800:2007,  cl 8.2.
        """

        return bool(M <= M_d)


    @staticmethod
    def cl_8_2_Design_bending_strength_of_laterally_unsupported_beam(z_p, z_e, f_y, v, v_d, m_dv, plastic=False,
                                                                      compact=False):
        """Calculation of bending strength of laterally unsupported beam for low shear  and high shear case
            Args:
                beta_b - 1 for plastic and compact
                         Z_e/Z_p for semi-compact
                z_e - Elastic section modulus of the cross section in mm**3
                z_p - Plastic section modulus of the cross section in mm**3
                f_y - yield stress of the material (in N/ mm**2 )
                v  - Factored design shear strength in N
                v_d - Design shear strength in N
                m_dv - Design bending strength under high shear as defined in Cl 9.2 in N*m
```

```
                plastic - True if beam is plastic else False
                compact - True for compact section else False

            Returns:
                m_d - Design Bending  strength in N*m
            Note:
                References:
                IS800:2007,  cl 8.2.1.2, cl. 8.2.1.3

        """
        beta_b = z_e / z_p #semi-compact section
        if plastic is True:
            beta_b = 1
        if compact is True:
            beta_b = 1
        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        if v <= 0.6 * v_d:
            m_d = beta_b * z_p * f_y / gamma_m0


        if v > 0.6 * v_d:
            m_d = m_dv
        return m_d

    # cl8.2.2 DESIGN BENDING STRENGTH OF LATERALLY UNSUPPORTED BEAMS
    # cl8.2.2.1 Elastic lateral torsional buckling moment
    @staticmethod
    def
cl_8_2_2_1_Elastic_lateral_torsional_buckling_moment_doubly_symmetric(
I_t, I_w, I_y, E, G, L_LT):
        """
            Calculation of elastic critical moment of lateral torsional
buckling for simply supported, prismatic members
            with symmetric c/s
            Args:

                I_t - torsional constant
                I_w - warping constant
                I_y - moment of inertia about weaker axis
                E - Young's Modulus of Elasticity
                G - modulus of rigidity
                L_LT - effective length for lateral torsional buckling
            Return:
                M_cr - Elastic lateral torsional buckling moment in N*mm
            Notes:
                Reference:
                IS 800:2007, cl. 8.2.2.1.

        """


        M_cr = math.sqrt((math.pi ** 2 * E * I_y / L_LT ** 2) * (G * I_t +
math.pi ** 2 * E * I_w / L_LT ** 2))

        return M_cr


    @staticmethod
    def cl_8_2_2_design_bending_strength_of_laterally_unsupported_beam(Z_p,
 Z_e, L_LT, f_y, I_y,  I_t, I_w,  E, G,section,plastic=False, compact=
False):
```

```
    """
          Calculation of design bending strength of laterally unsupported beam
          Args:
              Z_p - plastic section modulus with respect to extreme compression fibre
              Z_e - elastic section modulud with respect to extreme compression fibre
              L_LT - effective length for lateral torsional buckling
              I_y - moment of inertia about minor axis of c/s
              f_y - yield stress
              I_t - torsional constant
              I_w - warping constant
              E - modulus of elasticity
              G - modulus of rigidity
              plastic - boolean True if section is plastic
              compact - boolean True if section is compact
              section - Either 'Rolled_steel_section' or "Welded_steel_section'

        Returns:
            M_d - Design bending strength of laterally unsupported beam in N*mm

        Note:
            Reference:
            IS 800:2007, cl. 8.2.2.

    """
    beta_b = Z_p / Z_e
    if plastic is True:
        beta_b = 1
    if compact is True:
        beta_b = 1

    if section == 'Rolled_steel_section':
        alpha_LT = 0.21
    elif section == 'Welded_steel_Connection':
        alpha_LT = 0.49

    M_cr = IS800_2007.cl_8_2_2_1_Elastic_lateral_torsional_buckling_moment_doubly_symmetric(I_t, I_w, I_y, E, G, L_LT)
    f_cr_b = M_cr / (beta_b * Z_p)
    Lambda_LT = min(math.sqrt(f_y / f_cr_b), math.sqrt(1.2 * Z_e * f_y / M_cr))
    phi_LT = 0.5 * (1 + alpha_LT * (Lambda_LT - 0.2) + Lambda_LT ** 2)
    X_LT = min(1.0, 1 / (phi_LT + math.sqrt(phi_LT ** 2 - Lambda_LT ** 2)))


    if Lambda_LT < 0.4:
       X_LT = 1


    gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']

    f_bd = X_LT * f_y / gamma_m0

    M_d = beta_b * Z_p * f_bd

```

```
        return M_d


    # cl8.4 Shear Design
    @staticmethod
    def cl_8_4_design_shear_strength_of_beam(V_n):
        """
            Design shear strength
        Args:
            V_n -Nominal  shear strength of a cross-section
        Return
            V_d - Design shear strength in N
        Note:
            Reference:
            IS 800:2007, cl. 8.4

        """

        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        V_d = V_n / gamma_m0
        return V_d


    @staticmethod
    def cl_8_4_1_nominal_plastic_shear_resistance_under_pure_shear(A_v,
   f_yw):
        """
            Calculation of nominal plastic shear resistance under pure
   shear

            Args:

                A_v - Shear area
                f_yw - yield shear of the web
            Returns:
                V_n - Nominal plastic shear resistance under pure shear
            Note:
                Reference:
                IS 800:2007, cl. 8.4.1
        """

        V_n = A_v * f_yw / math.sqrt(3)
        return V_n



    def cl_8_4_1_1_shear_area_of_different_section(A, b, d, h, t_f, t_w,
   section, Axis_of_Bending, load_application_axis,
                                                     cross_section):
        """
            Calculation of shear area of different section

            Args:
                A - cross section area in mm**2
                b - overall breadth of tubular section,breadth of I -
   section flange in mm
                d - clear depth of the web between flange in mm
                h- overall depth of the section in mm
                t_f - thickness of the flange in mm
                t_w - thickness of the web in mm
                section - Either 'I section ' and 'Channel Section' or '
```

```
Rectangular hollow section of uniform depth'
                        or 'Circular hollow tubes of uniform thickness' or
'plates' or 'solid bars'
                Load_application_axis - Either 'Loaded parallel to depth'
or 'Loaded parallel to width(b)'
                Axis_of_Bending - Either 'Major Axis Bending' or 'Minor
Axis Bending'
                cross_section - Either 'Hot Rolled' or ' Welded'


            Return:
                A_v - Shear area in mm*mm

            Note:
                Reference:
                IS 800:2007, cl. 8.4.1.1

        """
        if section == 'I section' or section == 'Channel Section':
            if Axis_of_Bending == 'Major Axis Bending':
                if cross_section == 'Hot-Rolled':
                    A_v = h * t_w
                else:
                    A_v = d * t_w
            return A_v

        if Axis_of_Bending == 'Minor Axis Bending':
            if cross_section == 'Hot-Rolled' or cross_section == 'Welded':
                A_v = 2 * b * t_f
            return A_v
        if section == 'Rectangular hollow section of uniform depth':
            if load_application_axis == 'Loaded parallel to depth':
                A_v = A * h / (b + h)
            else:
                A_v = A * b / (b + h)
            return A_v

        if section == 'Circular hollow tubes of uniform thickness':
            A_v = A
            return A_v

        if section == 'plates' or section == 'solid':
            A_v = A
            return A_v


    # cl8.4.2 TODO: CHECK RESISTANCE TO SHEAR BUCKLING
    # cl8.4.2.1 Check for resistance to shear buckling
    @staticmethod
    def cl_8_4_2_shear_buckling_check(d, t_w, k_v, fy, stiffeners):
        """
            Check for resistance against shear buckling
            Args:
                d - clear depth of web between flanges
                t_w - thickness of web
                k_v -shear buckling coefficient
                fy - yield stress in N/mm^2
                stiffeners - True if web has stiffeners else False
            Return:
                check - True if check is satisfied else false
            Note:
```

```
            Reference:
            IS 800:2007, cl. 8.4.2.1
        """
        epsilon = math.sqrt(250/fy)
        if not stiffeners:
            val = 67 * epsilon

        else:
            val = 67 * epsilon * math.sqrt(k_v/5.35)
        if d / t_w > val :
            check = True
        else:
            check = False

        return check

    # cl8.4.2.2 Shear buckling design method
    @staticmethod
    def cl_8_4_shear_buckling_coeff_Kv(only_at_support, c=None, d=None):
        """
        Args:
            only_at_support - True if transverse stiffeners are provided
    only at support
                                  else False
            c -  spacing of transverse stiffeners
            d -  depth of web
        Returns:
            k_v - shear buckling coefficient
        Note:
              Reference - IS800_2007 cl.8.4.2.1 and cl.8.4.2.2
        """
        if only_at_support:
            k_v = 5.35
        elif c / d < 1:
            k_v = 4 + 5.35 / (c / d) ** 2
        else:
            k_v = 5.35 + 4 / (c / d) ** 2
        return k_v

    @staticmethod
    def cl_8_4_2_2_nominal_shear_post_critical(A_v,k_v,mu,E,d,t_w,f_yw):
        """
        Calculates nominal shear strength as governed by buckling using
    simple post critical method
        Args:
            A_v - shear area defined in cl 8.4.1.1
            k_v - shear buckling coefficient
            mu - poisson's ratio
            E - modulus of elasticity
            d - depth of web
            t_w - thickness of web
            f_yw - characteristic yield stress of web material
        Return:
            V_n - nominal shear strength
        Note:
             Reference: IS800:2007 cl.8.4.2.2
        """
        T_cr_c = (k_v * math.pi ** 2 * E) / (12 * (1 - mu ** 2) * (d / t_w)
     ** 2)
        lambda_w = math.sqrt(f_yw / (math.sqrt(3) * T_cr_c))

```

```
        if lambda_w < 0.8:
            T_b =  (f_yw / math.sqrt(3))
        elif 0.8<lambda_w < 1.2:
            T_b = (1 - 0.8 * max((lambda_w - 0.8),0)) * (f_yw / math.sqrt(3))
        else:
            T_b = f_yw / (math.sqrt(3) * lambda_w ** 2)

        V_cr = A_v * T_b
        V_n = V_cr
        return V_n

    @staticmethod
    def cl_8_4_2_2_nominal_shear_tension_field(A_v,d,t_w,t_f,b_f,f_yw,c,f_yf,N_f):
        """
        Calculates nominal shear strength as governed by buckling using tension field method
        Args:
            A_v - shear area defined in cl 8.4.1.1
            k_v - shear buckling coefficient
            mu - poisson's ratio
            E - modulus of elasticity
            d - depth of web
            t_w - thickness of web
            t_f - thickness of flange
            b_f - width of flange
            f_yw - characteristic yield stress of web material
            c - spacing of stiffeners in web
            f_yf - characteristic yield stress of flange material
            N_f - axial force in flange due to to overall bending and external axial force
        Return:
            Note: Reference - IS800:2007 cl.8.4.2.2
        """
        if c/d < 1.0:
            return 'error : c/d must be greater than or equal to 1'

        phi = math.atan(d / c) / 1.5
        psi = 1.5 * T_b * math.sin(2 * phi)

        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        M_fr = 0.25 * b_f * t_f ** 2 * f_yf * (1 - (N_f / (b_f * t_f * f_yf / gamma_m0)) ** 2)

        s = min(c, (2 / math.sin(phi) * math.sqrt(M_fr / f_yw * t_w)))
        s_c = s
        s_t = s
        w_tf = d * math.cos(phi) - (c - s_c - s_t) * math.sin(phi)
        f_v = math.sqrt(f_yw ** 2 - 3 * T_b ** 2 + psi ** 2) - psi
        V_p = A_v * f_yw / math.sqrt(3)
        V_tf = min(V_p, (A_v * T_b + 0.9 * w_tf * t_w * f_v * math.sin(phi)))
        V_tf = V_n
        return V_n



    # Stiffened web Panels
    # End plate Design
    # cl8.5.3 Anchor forces
```

```
    @staticmethod
    def cl_8_5_3_anchor_forces(d, t, f_y, V, V_cr, V_tf):
        """ Calculation of resultant longitudinal shear and moment
            Args:
                d - web depth in mm
                t - thickness of the section in mm
                f_y - yield stress in N/mm**2
                V_cr - critical shear strength as defined in cl8.4.2.2.
                V_tf - basic shear strength as defined in cl8.4.2.2.

                V - actual factored shear force
            Return:
                M_tf - resultant longitudinal moment in N*mm
                R_tf - resultant longitudinal shear in N*mm

            Note:
                Reference:
                IS 800:2007, cl. 8.5.3
        """

        V_p = d * t * f_y / math.sqrt(3)

        H_q = 1.25 * V_p * math.sqrt(1 - V_cr / V_p)

        if V < V_tf:
            H_q *= (V - V_cr) / (V_tf - V_cr)


        R_tf = H_q / 2
        M_tf = H_q * d / 10

        return (R_tf, M_tf)

    # cl8.6 Design of Beams and Plate Girders with Solid Webs
    # cl8.6.1Minimum Web Thickness
    # cl8.6.1.1 Serviceability requirement
    @staticmethod
    def cl_8_6_1_1_minimum_web_thickness(d, t_w, c, f_yw, serviceability_requirement, web_connection_to_flange):
        """
            Checking the serviceability requirement of  minimum thickness of web
            Args:
                d - web depth in mm
                t_w - thickness of web in mm
                c - spacing of transverse stiffener in mm
                epsilon_w - yield stress ratio of web
                f_yw - yield stress of the web in N/mm**2
                serviceability_requirement - 'transverse_stiffener_not_provided',
                                             'only_transverse_stiffeners_provided_in_web_flange_connection_along_both_longitudinal_ed
'
                                             'transverse_and_longitudinal_stiffener_at_one_level_as_cl_8_7_13'
                                             'second_longitudinal_stiffener_provided_at_NA'
                web_connection_to_flange - 'along_both_longitudinal_edges'
                                           'along_one_longitudinal_edge'
>
            Return:
```

```
            True, if safety condition is satisfied else False
        Note:
            Reference:
            IS 800:2007, cl. 8.6.1.1

    """
    epsilon_w = math.sqrt(250 / f_yw)

    if serviceability_requirement == 'transverse_stiffener_not_provided
':
        if web_connection_to_flange == 'along_both_longitudinal_edges':
            return d/t_w <= 200 * epsilon_w
        elif web_connection_to_flange == 'along_one_longitudinal_edge':
            return d/t_w <= 90 * epsilon_w
    elif serviceability_requirement == '
only_transverse_stiffeners_provided_in_web_flange_connection_along_both_longitudinal_ed
':
        if 3 * d >= c >= d:
            return d / t_w <= 200 * epsilon_w
        elif 0.74 * d <= c < d:
            return c / t_w <= 200 * epsilon_w
        elif c < 0.74 * d:
            return d / t_w <= 270 * epsilon_w
        else:
            return 'web_is_unstiffened'

    elif serviceability_requirement == '
transverse_and_longitudinal_stiffener_at_one_level_as_cl_8_7_13':
        if d < c <= 2.4 * d:
            return bool(d / t_w <= 250 * epsilon_w)
        elif 0.74 * d <= c < d:
            return bool(c / t_w <= 250 * epsilon_w)
        else:
            return bool(d / t_w <= 340 * epsilon_w)

    else:
        return bool( d / t_w <= 400 * epsilon_w)

# cl 8.6.1.2 Compression flange buckling requirement
def cl_8_6_1_2_web_thickness_to_avoid_buckling_of_compression_flange(d,
 t_w, c, f_yf, Transverse_stiffener):
    """
        Check for minimum web thickness to avoid buckling of
compression flange
        Args:
            d - depth of the web in mm
            t_w - thickness of the web in mm
            c - spacing of transverse stiffener in mm
            epsilon_f - yield stress ratio of flange
            f_yw - yield stress of compression flange in N/mm**2
            Transverse_stiffener - either 'provided' or 'not provided'
        Return:
            True or False
        Note:
            Reference:
            IS 800:2007, cl. 8.6.1.1

    """
    epsilon_f = math.sqrt(250 / f_yf)
    if Transverse_stiffener == 'not provided':
       return bool( d / t_w <= 345 * epsilon_f ** 2)
```

```python
        else:
            if c >= 1.5 * d:
                return bool(d / t_w <= 345 * epsilon_f ** 2)
            else:
                return bool(d / t_w <= 345 * epsilon_f)

    # cl8.7.1.5 Buckling resistance of stiffeners
    # Effective length for load carrying web stiffeners
    @staticmethod
    def cl_8_7_1_5_effective_length_for_load_carrying_web_stiffeners(L, restrained_condition):
        """
            Calculation of Effective length for load carrying web stiffeners for calculating
            buckling resistance F_xd
            Args:
                L - length of stiffener in mm
                restrained_condition - Either 'flange_restrained_against_rotation' or
                                          'flange_not_restrained_against_rotation'
            Returns:
                K_L - effective length for load carrying web stiffeners in mm
            Note:
                Reference:
                IS 800:2007,   cl 8.7.1.5
        """


        if restrained:
            K_L = 0.7 * L
            return K_L
        else:
            K_L = L
            return K_L

    # Cl 8.7.2.4 Minimum stiffeners
    @staticmethod
    def cl_8_7_2_4_I_s_for_transverse_web_Stiffeners_not_subjected_to_external_load(c, d, t_w):
        """
            Calculation of second moment of area when transverse web stiffener
                not subjected to external load
            Args:
                d - depth of thw web in mm
                t_w - minimum required web thickness foe spacing using tension filed action ,as given in cl8.4.2.1 in mm
                c - actual stiffener spacing in mm
            Return:
                I_s_min - second moment of area in mm**4
            Note:
                Reference:
                IS 800:2007,  cl 8.7.2.4
        """
        if c / d >= math.sqrt(2):
            I_s_min = 0.75 * d * t_w ** 3
        else:
            I_s_min = 1.5 * d ** 2 * t_w ** 3 / c ** 2
```

```python

        return I_s_min

    # cl 8.7.2.5 Buckling check on intermediate transverse web stiffeners
    def cl_8_7_2_5_buckling_check_on_intermediate_transverse_web_stiffener(V, F_qd, F_x, F_xd, M_q, M_yq,V_cr):
        """
            Buckling check on intermediate transverse web stiffeners
        Args:
                F_qd - design resistance of the intermediate stiffeners in N
                V -factored shear force adjacent to the stiffener in N
                F_qd - deign resistance of an intermediate web stiffener
                        to buckling corresponding to buckling about at
                         axis parallel to the web as in cl 8.7.1.5 in N
                F_x - external load or reaction at the stiffener in N
                F_xd - design resistance of a load carrying stiffener
                        corresponding to buckling about axis parallel
                        to the web as in cl 8.7.1.5 in N
                M_q - moment on the stiffener due to eccentrically
                        applied load anf transverse load, if any in N*mm
                M_yq - yield moment capacity og the stiffener based
                        on its elastic modulus about its centroidal
                        axis parallel to the web in N*mm
        Return:
            F_q - stiffener force in N

        Note:
            Reference:
            IS 800:2007,  cl 8.7.2.5, cl.8.7.3

        """
        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        F_q = min(V - V_cr / gamma_m0, F_qd)
        if (F_q - F_x) / F_qd + F_x / F_xd + M_q / M_yq <= 1:
            return F_q
        else:
            return 'cl.8.7.2.5.warning:buckling check on intermediate transverse web stiffener not satisfied'


    # cl 8.7.2.6 Connection of intermediate stiffeners to web
    @staticmethod
    def cl_8_7_2_6_shear_between_each_component_of_stiffener_and_web(t_w, b_s,s_e):

        """
            Calculation of minimum allowable shear between each component of stiffener and web
            Args:
                t_w - web thickness in mm
                b_s - outstanding  width of the stiffeners in mm

            Returns:
                V_is_min - minimum shear between each component of stiffener and web in N/mm


               Notes:
                Reference:
                IS 800:2007,  cl 8.7.2.6
```

```
        """
        V = t_w ** 2 / 5 * b_s + s_e
        return V


    # cl 8.7.3 Load Carrying stiffeners
    # cl 8.7.3.1 Web Checking
    @staticmethod
    def cl_8_7_3_1_area_of_cross_section_of__web(b_1, n_1, t_w):
        """
            Calculation of area of cross section of the web
            Args:
                b_1 - width of stiff bearing on the flange in mm
                n_1 - dispersion of the load through the web
                      45 degree, to the level of half the depth
                      of the cross section
                t_w - web thickness in mm
                       45 degrees, to the level of half the depth
                       of the cross section
            Returns:
                A_w - area of cross section of the web in mm**2
            Notes:
                Reference:
                IS 800:2007,  cl 8.7.3.1
        """
        A_w = (b_1 + n_1) / t_w
        return A_w

    # cl 8.7.4 Bearing Stiffeners

    @staticmethod
    def cl_8_7_4_force_applied_through_flange_by_loads_(b_1, n_2, t_w, f_yw
    ):
        """
            Calculation of force applied through a flange by load or
    reaction
            exceeding the local capacity of the web at its connection

            Args:
                b_1 - stiff bearing length in mm
                n_2 - length obtained by dispersion through the flange to
    the web
                            junction at a slope of 1:2.5 to the plane of the
    flange  in mm
                t_w - thickness of the web in mm
                f_yw - yield stress of the web in N/mm**2

            Returns:
                F_w - force applied through flange by loads  in N

            Notes:
                Reference:
                IS 800:2007,  cl 8.7.4

        """
        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        F_w = (b_1 + n_2) * t_w * f_yw / gamma_m0
        return F_w

    # cl 8.7.5 Design of load carrying stiffeners
```

```
    # cl 8.7.5.1 Buckling check
    # cl 8.7.5.2 Bearing check

    @staticmethod
    def cl_8_7_5_2_bearing_strength_of_stiffeners(F_x, A_q, f_yq):
        """
            Calculation of bearing strength of stiffeners

            Args:
                F_x - external load or reaction in N
                A_q - area of the stiffeners in contact with the flange in mm
                f_yq - yield stress of the stiffeners in N/mm**2

            Return:
                F_psd - bearing strength of stiffeners in N

            Notes:
                Reference:
                IS 800:2007,  cl 8.7.4
        """
        gamma_m0 = IS800_2007.cl_5_4_1_Table_5["gamma_m0"]['yielding']
        F_psd = max(A_q * f_yq / (0.8 * gamma_m0), F_x)
        return F_psd

    # cl 8.7.9 Torsional Stiffeners
    @staticmethod
    def cl_8_7_9_minimum_second_moment_of_area_of_the_stiffener_section(D, T_cf, L_LT, r_y):
        """
            calculation of  minimum second moment of area of the stiffener

            Args:
                D = overall depth of beam at support in mm
                T_cf = maximum thickness of compression flange in the span under consideration in mm
                K_L=  laterally unsupported effective length of the compression flange of the beam in mm
                r_y = radius of gyration of the beam about the minor axis in mm

            Returns:
                I_s_min = calculation of  minimum second moment of area of the stiffener in mm**4


            Notes:
                Reference:
                IS 800:2007,  cl 8.7.4
        """
        if L_LT / r_y <= 50:
            alpha_s = 0.006
        elif 50 < L_LT / r_y <= 100:
            alpha_s = 0.3 / (L_LT / r_y)
        else:
            alpha_s = 30 / (L_LT / r_y) ** 2

        I_s_min = 0.34 * alpha_s * D ** 3 * T_cf

        return I_s_min

```

```
    # ========================================
```

Appendix D

Python Code for Annex E IS: 800 2007

```python
"""    ANNEX  E       ELASTIC LATERAL TORSIONAL BUCKLING   """
import math
# Table 5 Partial Safety Factors for Materials, gamma_m (dict)
class IS800_2007(object):
    cl_5_4_1_Table_5 = {"gamma_m0": {'yielding': 1.10, 'buckling': 1.10},
                        "gamma_m1": {'ultimate_stress': 1.25},
                        "gamma_mf": {'shop': 1.25, 'field': 1.25},
                        "gamma_mb": {'shop': 1.25, 'field': 1.25},
                        "gamma_mr": {'shop': 1.25, 'field': 1.25},
                        "gamma_mw": {'shop': 1.25, 'field': 1.50}
                        }
    # CALCULATION OF EFFECTIVE LENGTH AGAINST LATERAL TORSIONAL BUCKLING.
    def cl_8_3_1_Effective_length_for_simply_supported_beams(L, D, Restraint_Condition_1, Restraint_Condition_2,
    Loading_Condition):

        """
            Calculate effective length against lateral torsional buckling for simply supported Beams and girders
            where no lateral restraint  to the compression falnge is provided as per cl.8.3.1

            Args:
                L -  Span of simply suppotred beams and girders in mm (float)
                D -  Overall depth of he beam in mm (float)

                Restraint_Condition - Either "Torsional Restraint" or "wraping Restraint"
                Restraint_Condition_1- "Torsional Restraint"
                Restraint_Condition_2- "Warping_Restraint"

                "Torsional Restrained" - Either "Fully_resrtrained" or
                                        "Partially_restrained_by_bottom_flange_support_condition" or
                                        "Partially_restrained_by_bottom_flange_support_condition"

                "Warping_Restraint" - Either "Both_flange_fully_restrained" or
                                           "compression_flange_fully_restrained"
```

```
    or
                                    "
    Compression_flange_partially_restrained" or
                                    "Warping_not_restrained_in_both_flange
    "


                Loading_Condition  - Either "Normal" or " Destabilizing"



            Returns:
                L_LT  - cl_8_3_1_Effective length for simply supported
    Beams in mm (float)


            Note:
                References:
                IS800:2007, Table 15 (cl 8.3.1)

        """

        if Restraint_Condition_1 == "Fully Restrained":
            if Restraint_Condition_2 == "Both flanges fully restrained":
                if Loading_Condition == "Normal":
                    return 0.70 * L
                else:
                    return 0.85 * L
            if Restraint_Condition_2 == "Compression flange fully
    Restrained":
                if Loading_Condition == "Normal":
                    return 0.75 * L
                else:
                    return 0.90 * L
            if Restraint_Condition_2 == "Both flanges partially restrained"
    :
                if Loading_Condition == "Normal":
                    return 0.80 * L
                else:
                    return 0.95 * L
            if Restraint_Condition_2 == "Compression flange partially
    Restrained":
                if Loading_Condition == "Normal":
                    return 0.85 * L
                else:
                    return 1.00 * L
            if Restraint_Condition_2 == "Wraping not restrained in both
    flanges":
                if Loading_Condition == "Normal":
                    return 1.00 * L
                else:
                    return 1.20 * L
        if Restraint_Condition_1 == "Partially restrained by bottom flange
    support connection":
            if Restraint_Condition_2 == "Wraping not restrained in both
    flages":
                if Loading_Condition == "Normal":
                    return 1.00 * L + 2 * D
                else:
                    return 1.20 * L + 2 * D
        if Restraint_Condition_1 == "Partially restrained by bottom flage
```

```
bearing support":
        if Restraint_Condition_2 == "Wraping not restrained in both flages":
            if Loading_Condition == "Normal":
                return 1.2 * L + 2 * D
            else:
                return 1.4 * L + 2 * D

# Effective length for cantilever Beam

def cl_8_3_3_Table_16_Efective_length_for_cantilever_beam(L, D, Restraint_Condition_1, Restraint_Condition_2,
Loading_condition):
    """
        Calculate effective length for catiliver beam of projecting length L as per cl.8.3.3

        Args:
            L - Projecting Length of cantiliver beam in mm (float)
            D -  Overall depth of he beam in mm (float)

            Restrained_condition - Either "At support" or "At Top"

            Restraint_Condition_1- "At support"
            Restraint_Condition_2- "At Top"

            At_support -  Either "continous, with lateral restraint to top"
                          or "continous, with  partial torsional restraint"
                          or "continous, with lateral and tosional restraint "
                          or "Restrained laterally,torsionally and against rotation on plan "

            At_top - Either  "free"
                    or "lateral restraint to top flange"
                    or "Torsional restraint"
                    or  "Lateral and torsional restraint"

            Loading_condition - Either "Normal" or  "Destablizing"

        Returns:
            L_LT = cl_8_3_3_Table_16_Efective_length_for_cantiliver_beam
        Note:
            References:
            IS800:2007, Table 16 (cl 8.3.3)
    """

    if Restraint_Condition_1 == "Continuous, with lateral restraint to top flage":
        if Restraint_Condition_2 == "Free":
            if Loading_condition == "Normal":
                return (3.0 * L + 0 * D)
            else:
                return (7.5 * L + 0 * D)
        if Restraint_Condition_2 == "Lateral restraint to top flage":
            if Loading_condition == "Normal":
                return (2.7 * L + 0 * D)
```

```
                else:
                    return (7.5 * L + 0 * D)
            if Restraint_Condition_2 == "Torsional restraint":
                if Loading_condition == "Normal":
                    return (2.4 * L + 0 * D)
                else:
                    return (4.5 * L + 0 * D)
            if Restraint_Condition_2 == "Lateral and Torsional restraint":
                if Loading_condition == "Normal":
                    return (2.1 * L + 0 * D)
                else:
                    return (3.6 * L + 0 * D)
        if Restraint_Condition_1 == "Continuous,with partial torsional restraint":
            if Restraint_Condition_2 == "Free":
                if Loading_condition == "Normal":
                    return (2.0 * L + 0 * D)
                else:
                    return (5.0 * L + 0 * D)
            if Restraint_Condition_2 == "Lateral restraint to top flage":
                if Loading_condition == "Normal":
                    return (1.8 * L + 0 * D)
                else:
                    return (5.0 * L + 0 * D)
            if Restraint_Condition_2 == "Torsional restraint":
                if Loading_condition == "Normal":
                    return (1.6 * L + 0 * D)
                else:
                    return (3.0 * L + 0 * D)
                if Restraint_Condition_2 == " Lateral and Torsional restraint":
                    if Loading_condition == "Normal":
                        return (1.4 * L + 0 * D)
                    else:
                        return (2.4 * L + 0 * D)
        if Restraint_Condition_1 == "Continuous,with lateral and torsional restraint":
            if Restraint_Condition_2 == "Free":
                if Loading_condition == "Normal":
                    return (1.0 * L + 0 * D)
                else:
                    return (2.5 * l + 0 * D)
            if Restraint_Condition_2 == "Lateral restraint to top flage":
                if Loading_condition == "Normal":
                    return (0.9 * L + 0 * D)
                else:
                    return (2.5 * L + 0 * D)
            if Restraint_Condition_2 == "Torsional restraint":
                if Loading_condition == "Normal":
                    return (0.8 * L + 0 * D)
                else:
                    return (1.5 * L + 0 * D)
            if Restraint_Condition_2 == " Lateral and Torsional restraint":
                if Loading_condition == "Normal":
                    return (0.7 * L + 0 * D)
                else:
                    return (1.2 * L + 0 * D)
        if Restraint_Condition_1 == "Restrained laterally,torsionally and against rotation on plan":
            if Restraint_Condition_2 == "Free":
                if Loading_condition == "Normal":
```

```
                return (0.8 * L + 0 * D)
            else:
                return (1.4 * L + 0 * D)
        if Restraint_Condition_2 == "Lateral restraint to top flage":
            if Loading_condition == "Normal":
                return (0.7 * L + 0 * D)
            else:
                return (1.4 * L + 0 * D)
        if Restraint_Condition_2 == "Torsional restraint":
            if Loading_condition == "Normal":
                return (0.6 * L + 0 * D)
            else:
                return (0.6 * L + 0 * D)
        if Restraint_Condition_2 == "Lateral and Torsional restraint":
            if Loading_condition == "Normal":
                return (0.5 * L + 0 * D)
            else:
                return (0.5 * L + 0 * D)

def cl_8_3_Effective_length_against_torsional_restraint(L, D, Beam_type
, Restraint_Condition_1,

Restraint_Condition_2, Loading_Condition):
    """
        Calculation of effective length for given type of beam type as
per cl.8.3

    Args:
        L-  Span of simply suppotred beams and girders in mm (float)
for
            "
Simply_supported_with_no_lateral_restrained_to_the_compression_flanges
",
             Projecting Length of cantiliver beam in mm (float) for
             "Cantilever_beam",
             Length of relevent segment between the lateral restraint
in mm (float) for
             "Simply_supported_with_intermediate_lateral_restraints",
             Centre-to-centre distance of the restraint member in mm (
float) for
             "
Beam_provided_with_members_to_give_effective_lateral_restrain_to_compression_flange_at_
"

        D -  Overall depth of he beam in mm (float)


        Beam_type - Either "
Simply_supported_with_no_lateral_restrained_to_the_compression_flanges"
                    or "
Simply_supported_with_intermediate_lateral_restraints"
                    or "
Beam_provided_with_members_to_give_effective_lateral_restrain_to_compression_flange_at_
"
                    or "Cantilever_beam"

        FOR "
Simply_supported_with_no_lateral_restrained_to_the_compression_flanges"

        Restraint_Condition - Either "Torsional Restraint" or "wraping
Restraint"
```

```

        Restraint_Condition_1- "Torsional Restraint"
        Restraint_Condition_2- "Warping_Restraint"

        "Torsional Restrained" - Either "Fully_resrtrained" or
                                    "Partially_restrained_by_bottom_flange_support_condition" or
                                    "Partially_restrained_by_bottom_flange_support_condition"

        "Warping_Restraint" - Either "Both_flange_fully_restrained" or
                                 "compression_flange_fully_restrained" or
                                 "Compression_flange_partially_restrained" or
                                 "Warping_not_restrained_in_both_flange"


        FOR "Cantilever_beam"

        Restrained_condition - Either "At support" or "At Top" for "Cantilever_beam"

        Restraint_Condition_1- "At support"
        Restraint_Condition_2- "At Top"

        At_support -  Either "continous, with lateral restraint to top"
                          or "continous, with  partial torsional restraint"
                          or "continous, with lateral and tosional restraint "
                          or "Restrained laterally,torsionally and against rotation on plan "

        At_top - Either  "free"
                    or "lateral restraint to top flange"
                    or "Torsional restraint"
                    or  "Lateral and torsional restraint"

        Loading_condition - Either "Normal" or  "Destablizing"



    Returns :
        L_LT - Effective_length_of_beam in m (float)

    Note:
            References:
            IS800:2007,  cl 8.3.

    """

    if Beam_type == "Simply_supported_with_no_lateral_restrained_to_the_compression_flanges":
        L_LT = cl_8_3_1_Effective_length_for_simply_supported_beams(L, D, Restraint_Condition_1,
                                                                    Restraint_Condition_2, Loading_Condition)
    elif Beam_type == "
```

```
Simply_supported_with_intermediate_lateral_restraints":
            L_LT = 1.2 * L
        elif Beam_type == "
Beam_provided_with_members_to_give_effective_lateral_restrain_to_compression_flange_at_
":
            L_LT = 1.2 * L
        else:
            L_LT = cl_8_3_3_Table_16_Efective_length_for_cantilever_beam(L,
 D, Restraint_Condition_1,

Restraint_Condition_2, Loading_Condition)

        return L_LT

        # E-1 ELASTIC CRITICAL MOMENT

    # E-1.1 General
    # TODO:Calculate L_LT = L_LT =
cl_8_3_Effective_length_against_lateral_torsional_buckling(L,D,
Beam_type,Restraint_Condition_1,Restraint_Condition_2,Loading_Condition
)

    def
Annex_E_1_1_elastic_critical_moment_corresponding_to_lateral_torsional_buckling_of_doub
(
            I_y, I_w, I_t, G, E, L_LT):

        """
            Calculate the elastic critical moment corresponding to lateral
                torsional buckling of a doubly symmetric prismatic beam
subjected to uniform
                moment in the unsupported length and torsionally
restraining lateral supports as per Annex E-1.1

            Args:
                I_y - Moment of inertia about the minor axis (in quartic mm
) (float)
                I_w - Warping constant of the cross- section (mm**4)(float)
                I_t - St. Venants torsion constant of the cross-section (mm
**4)(float)
                G -   Modulus of rigidity (float)
                L_LT =
cl_8_3_Effective_length_against_lateral_torsional_buckling(L,D,
Beam_type,Restraint_Condition_1,Restraint_Condition_2,Loading_Condition
)


            Returns:
                M_cr - Elastic critical moment corresponding to lateral
torsional
                        buckling of doubly symmetric prismatic beam ( in N*
mm) (float)


            Note:
                Reference:
                IS800:2007, Annex - E-1.1

        """

        sum_value = (I_w / I_y) + (G * I_t * L_LT * L_LT) / (pi * pi * E *
```

```
    I_y)

        M_cr = ((pi * pi * E * I_y) / (L_LT * L_LT)) * ((sum_value) ** 0.5)

        return M_cr

    def Annex_E_Table_42_constant_c_1_c_2_c_3(Loading_and_Support_condition
    , si, K):
        """
            Calculate Value of constant c_1,c_2,c_3 as per Table_42 Annex-E
            Args:
                Loading_and_Support_condition - Either "Simply supported
    with ends moments (M,si*M)"
                                                or "simply Supported beam
    with UDL"
                                                or "Fixed support with UDL"
                                                or "Simply Supported beam
    with point load at centre"
                                                or "Fixed Supported beam
    with point load at centre"
                                                or "Simply Supported beam
    with point at L/4 distance from both ends"

                Bending_Moment_diagram- BMD for "Simply supported with ends
     moments (M,si*M)" by varying value of 'si' as-
                                                si - [
    +1,+3/4,+1/2,+1/4,0,-1/4,-1/2,-3/4,-1 ]
                                        BMD for "simply Supported beam with
     UDL"
                                        si - 0 (Assumed value)
                                        BMD for "Fixed support with UDL"
                                        si - 0 (Assumed value)
                                        BMD for "Simply Supported beam with
     point load at centre"
                                        si - 0 (Assumed value)
                                        BMD for "Fixed Supported beam with
    point load at centre"
                                        si - 0 (Assumed value)
                                        BMD for "Simply Supported beam with
     point at L/4 distance from both ends"
                                        si - 0 (Assumed value)

                K - [1.0,0.7,0.5]

            Returns:
                Value of constant c_1,c_2,c_3

            Note:
                References:
                IS800:2007,Table 42,cl E-1.2
        """

        if Loading_and_Support_condition == "Simply supported with ends
    moments (M,si*M)":
            if si == 1:
                if K == 1.0:
                    return {"c1": 1.000, "c2": 0, "c3": 1.000}
                if K == 0.7:
                    return {"c1": 1.000, "c2": 0, "c3": 1.113}
                if K == 0.5:
                    return {"c1": 1.000, "c2": 0, "c3": 1.144}
```

```python
        if si == +3 / 4:
            if K == 1.0:
                return {"c1": 1.141, "c2": 0, "c3": 0.998}
            if K == 0.7:
                return {"c1": 1.270, "c2": 0, "c3": 1.565}
            if K == 0.5:
                return {"c1": 1.305, "c2": 0, "c3": 2.283}
        if si == +1 / 2:
            if K == 1.0:
                return {"c1": 1.323, "c2": 0, "c3": 0.992}
            if K == 0.7:
                return {"c1": 1.473, "c2": 0, "c3": 1.556}
            if K == 0.5:
                return {"c1": 1.514, "c2": 0, "c3": 2.271}
        if si == +1 / 4:
            if K == 1.0:
                return {"c1": 1.879, "c2": 0, "c3": 0.939}
            if K == 0.7:
                return {"c1": 2.092, "c2": 0, "c3": 1.473}
            if K == 0.5:
                return {"c1": 2.150, "c2": 0, "c3": 2.150}
        if si == 0:
            if K == 1.0:
                return {"c1": 1.563, "c2": 0, "c3": 0.977}
            if K == 0.7:
                return {"c1": 1.739, "c2": 0, "c3": 1.531}
            if K == 0.5:
                return {"c1": 1.788, "c2": 0, "c3": 2.235}
        if si == -1 / 4:
            if K == 1.0:
                return {"c1": 2.281, "c2": 0, "c3": 0.855}
            if K == 0.7:
                return {"c1": 2.538, "c2": 0, "c3": 1.340}
            if K == 0.5:
                return {"c1": 2.609, "c2": 0, "c3": 1.957}
        if si == -1 / 2:
            if K == 1.0:
                return {"c1": 2.704, "c2": 0, "c3": 0.676}
            if K == 0.7:
                return {"c1": 3.009, "c2": 0, "c3": 1.059}
            if K == 0.5:
                return {"c1": 3.093, "c2": 0, "c3": 1.546}
        if si == -3 / 4:
            if K == 1.0:
                return {"c1": 2.927, "c2": 0, "c3": 0.366}
            if K == 0.7:
                return {"c1": 3.009, "c2": 0, "c3": 0.575}
            if K == 0.5:
                return {"c1": 3.093, "c2": 0, "c3": 0.837}
        if si == -1:
            if k == 1.0:
                return {"c1": 2.752, "c2": 0, "c3": 0}
            if k == 0.7:
                return {"c1": 3.063, "c2": 0, "c3": 0}
            if k == 0.5:
                return {"c1": 3.149, "c2": 0, "c3": 0}
    if Loading_and_Support_condition == "simply Supported beam with UDL":
        if si == 0:
            if K == 1.0:
                return {"c1": 1.132, "c2": 0.459, "c3": 0.525}
```

```
            if K == 0.5:
                return {"c1": 0.972, "c2": 0.304, "c3": 0.980}
    if Loading_and_Support_condition == "Fixed support with UDL":
        if si == 0:
            if K == 1.0:
                return {"c1": 1.285, "c2": 1.562, "c3": 0.753}
            if K == 0.5:
                return {"c1": 0.712, "c2": 0.652, "c3": 1.070}
    if Loading_and_Support_condition == "Simply Supported beam with point load at centre":
        if si == 0:
            if K == 1.0:
                return {"c1": 1.365, "c2": 0.553, "c3": 1.780}
            if K == 0.5:
                return {"c1": 1.070, "c2": 0.432, "c3": 3.050}
    if Loading_and_Support_condition == "Fixed Supported beam with point load at centre":
        if si == 0:
            if K == 1.0:
                return {"c1": 1.565, "c2": 1.257, "c3": 2.640}
            if K == 0.5:
                return {"c1": 0.938, "c2": 0.715, "c3": 4.800}
    if Loading_and_Support_condition == "Simply Supported beam with point at L/4 distance from both ends":
        if si == 0:
            if K == 1.0:
                return {"c1": 1.046, "c2": 0.430, "c3": 1.120}
            if K == 0.5:
                return {"c1": 1.010, "c2": 0.410, "c3": 1.390}

# E-1.2 Elastic Critical Moment of a Section Symmetrical About Minor Axis
# TODO: Calculate c_1,c_2,c_3 = Annex_E_Table_42_constant_c_1_c_2_c_3(Loading_and_Support_condition,si,K)
def Elastic_critical_moment_of_a_section_symmetrical_about_minor_axis(L_LT, c_1, c_2, c_3, E, K, K_w, y_g, A_e, b,
                                                                      t, I_fc, I_ft, I_y, G, h, h_L, h_y, n,
                                                                      I_section=True, Open_section=True,
                                                                      Plain_flange=False):
    """
        Calculate the elastic critical moment for lateral torsional buckling for beam which is
        symmetrical only about the minor axis, and bending about major axis as per Annex E-1.2

        Args:
            c_1,c_2,c_3- Annex_E_Table_42_constant_c_1_c_2_c_3(Loading_and_Support_condition,si,K)
            K -  Effective length factors of the unsupported length accounting for boundry condition
                 at the end letral supports. It is analogus to the effective length factirs for
                 compression members with end rotational restraint.
            K_w -Warping restraint factor.
            y_g -y distance between the point of application of the load and the shear centre of
                 the cross-section and is positive when the load is acting towards the shear
```

```
                 centre from the point of application.
            y_s-  co-ordinate of the shear centre with respect to centriod,positive when the shear
                    centre is on the compression side of the centriod.
            y,z-  co-ordinate of the elemental area with respect to centriod of the section
            E -   Youngs modulus of elasticity  ( N per sq.mm)(float)
            G-     Modulus of regidity (float)
            I_y  - Moment of inertia about minor axis (in mm**4)(float)
            I_fc - Moment of inertia of the compression flange about minor axis of the entire section (in mm**4)(float)
            I_ft - Moment of inertia of the tension flange about minor axis of the entire section (in mm**4)(float)
            I_w  - The wraping constant either for " I_section_mono_symmetric_about_weak_axis" or for
                                                    "Angle,Tee, narrow_rectangle_section and approximetly for hollow_section"
            I_t - Torsion constant either "for open_section" or " hollow_section"
            A_e - Area encloed by the section (in sq mm)(float)
            b -   Breadth of the elements of the section(mm)(float)
            t -   Thickness of the elements of the section (mm)(float)
            h_L-  Height of the lip in mm(float)
            h-    overall height of the section in  mm(float)
            h_y - Distance between shear centre of the two flange of the cross-section in mm (float)
            Flange type - Either "Plain_flange" or "Lipped_flange"
            L_LT = cl_8_3_Effective_length_against_lateral_torsional_buckling(L,D,Beam_type,Restraint_Condition_1,Restraint_Condition_2,Loading_Condition)
        Returns:
            M_cr- Elastic_critical_moment_of_a_section_symmetrical_about_minor_axis (in N*mm)(float)

        Note:
            References:
            IS800:2007,cl E-1.2
    """

    beta_f = I_fc / (I_ft + I_fc)

    if I_section is True:
        I_w = (1 - beta_f) * beta_f * I_y * h_y * h_y
    else:
        I_w = 0

    if Plain_flange is True:
        if beta_f > 0.5:
            y_j = 0.8 * (2 * beta_f - 1) * h_y / 2.0
        else:
            y_j = 1.0 * (2 * beta_f - 1) * h_y / 2.0
    else:
        if beta_f > 0.5:
            y_j = 0.8 * (2 * beta_f - 1) * (1 + h_L / h_) * h_y / 2.0
        else:
            y_j = (2 * beta_f - 1) * (1 + h_L / h) * h_y / 2

    if Open_section is True:
        sum_value = 0
```

```
            for i in range(n - 1):
                sum_value += (b * t * t * t) / 3
                I_t = sum_value
        else:
            sum_value = 0
            for i in range(n - 1):
                sum_value += (b / t)

        I_t = 4 * A_e / sum_value

        T_1 = c_1 * (pi * pi * E * I_y) / (L_LT)
        T_2 = (K / K_w) ** 2 * (I_w / I_y)
        T_3 = (G * I_t * L_LT * L_LT) / (pi * pi * E * I_y)
        T_4 = ((c_2 * y_g) - (c_3 * y_j)) ** 2
        T_5 = ((c_2 * y_g) - (c_3 * y_j))

        M_cr = T_1 * (((T_2 + T_3 + T_4) ** 0.5) - T_5)

        return M_cr
    # =================================
```

Appendix E

Creating class for sectional properties of I section

```
import math

class I_sectional_Properties(object):
    def __init__(self,D,B,t_w,t_f,alpha=90,r_1=0,r_2=0):
        """ Calculation of sectional properties of built-up I section
        Args:
            D- depth of beam in mm
            B- Width of flange in mm
            t_w- thickness of web in mm
            t_f- thickness of flange in mm
            alpha- flange slope in degree
            r_1-root radius in mm
            r_2-toe radius in mm
        Return:
            A- Sectional Area of the section in mm**2
            M - Mass (kg/m)
            I_zz - second moment of area in cm**4
            I_yy - second moment of area in cm**4
            r_z - radius of gyration in cm
            r_y - radius of gyration in cm
            Z_ez - elastic modulus in cm**
            Z_ey - elastic modulus in cm**3
            Z_pz - plastic modulus in cm**3
            Z_py - plastic modulus in cm**3

        """

        self.A = ((2*B*t_f) + ((D-2*t_f)*t_w))/100
        self.M = 7850 * self.A / 10000
        self.I_zz = ((D - 2*t_f)**3 * t_w /12 + (B*t_f**3)/6+(B/2*t_f*(D-t_f)**2))/10000
        self.I_yy = ((D-2*t_f)*t_w**3 /12 + B**3*t_f/6)/10000
        self.r_z = math.sqrt(self.I_zz / self.A)
        self.r_y = math.sqrt(self.I_yy / self.A)
        self.Z_ez = (self.I_zz * 2*10) / (D)
        self.Z_ey = (self.I_yy * 2*10) / (B)
        self.y_p = (((D - 2*t_f)**2*t_w/8 + B*t_f*(D-t_f)/2) / ((D-t_f)/2*t_w + B*t_f ))/10
        self.Z_pz = (2 * (self.A / 2 * self.y_p))
        self.z_p = ((((D-2*t_f)*t_w**2)/8 + (B*t_f*B)/4)/((D-2*t_f)*t_w/2 + (B*t_f))
```

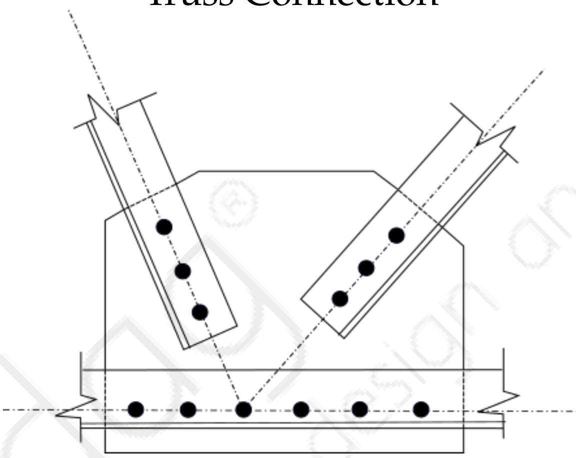
```
        self.Z_py = 2 * (self.A / 2 * self.z_p)
```

Appendix F

DDCL for Bolted Truss Connection

Design and Detailing Checklist (DDCL)
and
Design and Detailing Query (DDQ)

Truss Connection



Prepared by:
Rachna Gupta

Under the guidance of:
Prof. Siddhartha Ghosh

Indian Institute of Technology, Bombay

July 10, 2019

Contents

User Inputs

- Connecting members
  Truss Member Sections*
- Types of Connectivity*
  Bolted
- Factored loads
  Axial Force (kN)
- Material Property of section
  f_u (MPa)*
  f_y (MPa)*
- Fasteners Details
  Bolt Diameter (mm)*
  Bolt Type *
  Property class *
- Detailing
  Angle of inclination (degree)*
- Plate
  Thickness (mm)*
  f_u (MPa)*
  f_y (MPa)*

Design and Detailing Checks

Check 1

Bolt Design

The truss members and gusset plate are connected by bolts.

1.1 Gusset Plate

1.1.1 Gusset plate thickness (t_g)

Thickness of the gusset plate provided is usually equal to or slightly higher than members that are connected to gusset plate.
Thickness of gusset plate is calculated on the basis of maximum force acting on the truss members and is given by the table. [Reference:Table 5.12 Design of steel structure by N.Subramanian]

Maximum design force in diagonal(KN)	upto 200	200-450	450-750	750-1650	1650-2250	2250-3000
Thickness of gusset plate(mm)	8	10	12	16	18	20

1.2 Bolt Value

1.3 Bearing type bolts

1.3.1 Design strength of bolt (V_{db})

[Reference: Cl. 10.3.2, IS 800:2007]

The design strength of bolt is taken as the smaller of the value as governed by shear, V_{dsb} (1.3.2) and bearing V_{dpb} (1.3.3).

$$V_{db} = min\ (V_{dsb}, V_{dpb}) \tag{1.1}$$

1.3.2 Shear capacity (V_{dsb})

[Reference: Cl. 10.3.3, IS 800:2007]

Currently, conservatively assuming all the shear planes passes through the threads of the bolt. Shear capacity of bearing bolts,

$$V_{dsb} = \frac{f_u n_n A_{nb}}{\sqrt{3}\gamma_{mb}} \tag{1.2}$$

Where,
V_{dsb} = design strength of bolt, as governed by shear strength;

f_u = ultimate tensile strength of a bolt;
n_n = number of shear planes with threads intercepting the shear plane;
A_{nb} = net shear area of the bolt at threads,taken as the area corresponding to root diameter at the thread;
γ_{mb} = partial safety factor for bolt

1.3.3 Bearing capacity (V_{dpb})

[Reference: Cl. 10.3.4, IS 800:2007]

$$V_{dpb} = \frac{2.5\ k_b\ d\ t\ f_u}{\gamma_{mb}} \tag{1.3}$$

Where,
k_b is smaller of $\frac{e}{3\ d_0}, \frac{p}{3\ d_0} - 0.25, \frac{f_{ub}}{f_u}, 1.0$;
e, p = end and pitch distances of the bolt along line of action respectively;
d, d_0 = diameter of bolt and bolt hole respectively;
t = summation of the thicknesses of the connected plates experiencing bearing stress in the same direction;
γ_{mb} = partial safety factor.

1.4 Friction grip type bolts

1.4.1 Slip resistance (V_{dsf})

[Reference: Cl. 10.4.3, IS 800:2007]

$$V_{nsf} = \frac{F_o \mu_f n_e K_h}{\gamma_{mf}} \tag{1.4}$$

μ_f = coefficient of friction (slip factor);
n_e = number of effective interfaces offering frictional resistance to slip;
K_h = 1.0 for bolts in clearance holes and 0.85 for bolts in oversized holes;
F_o = proof load = $A_{nb} f_0$;
f_0 = proof stress $0.7 A_{ub}$;
γ_{mf} = partial safety factor.

1.5 Bolt Capacity

From here on, V_{bolt} is used where bolt capacity is considered. V_{bolt} is taken as V_{dsf} if friction grip bolt is considered and V_{db}, if bearing bolt is considered.

1.6 Check of long joint

[Reference: Cl. 10.3.3.1 IS 800 : 2007]

when $l_j \geq 15d$, design capacity will be reduced by factor β_{lj}.

$$\beta_{lj} = 1.075 - 0.005(l_j/d), 0.75 \leq \beta_{lj} \leq 1.0 \tag{1.5}$$

where,
l_j = length of joint (measured in the direction of load transfer);
d = nominal diameter of the bolt.

1.7 Check of large grip length

[Reference: Cl. 10.3.3.2 and Cl. 10.3.3.1 IS 800 : 2007]

when $l_g \geq 5d$, the design shear capacity is reduced by a factor β_{lg}.

$$\beta_{lg} = \frac{8}{3 + l_g/d} \tag{1.6}$$

where,
l_g = total thickness of the connected plates (grip length);
d = nominal diameter of the bolt.

Note: β_{lg} should not be greater than β_{lj}

1.8 Bolt capacity

$$V'_{db} = V_{bolt}\beta_{lj}\beta_{lg} \tag{1.7}$$

Check 2

Number of Bolts

2.1 Minimum number of bolts (n)

Number of bolts required for each truss member(n) is given by the equation;

$$n = \frac{P}{V_{bolt}} \tag{2.1}$$

If leg size of truss member section is than 130mm, then linear bolt raw will be provided. If leg size is greater than 130mm than either linear bolt raw or staggered bolt raw can be provided (chain bolting is usally not preferred as staggered bolting is more efficient).

2.2 Number of bolts in zero force member (n_0)

[Reference:Cl 10.7 IS 800: 2007]

When no axial force is acting on the truss member then a force of at least 0.3times the member design capacity acts at the ends of tensile or compression member and minimum number of bolt is calculated on this basis.

$$F_d = \frac{f_y A}{\gamma_{m0}} \tag{2.2}$$

where,
F_d = Member design capacity;

$$n_0 = \frac{0.3 F_d}{V_{bolt}} \tag{2.3}$$

Check 3

Gusset Plate Checks

3.1 Whitmore effective width (B_{eff})

[Reference: Page No. 365 Design of steel by structure N.Subramanian 13th Edition]

Maximum direct stress(compression or tension) in gusset plate from an individual member is estimated adequately by ensuring that the member force is distributed uniformly over an effective area given by 30° dispersion from outer row of fasteners as shown is figures:

3.1.1 For one bolt line

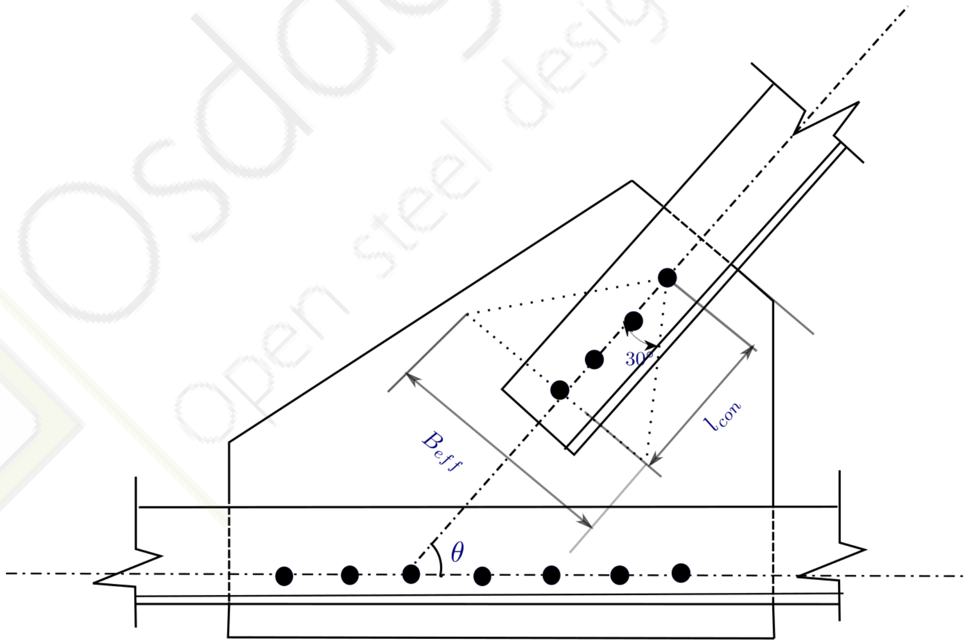


Figure 3.1: Whitmore Effective Width for Linear bolting arrangement

$$B_{eff} = 2l_{con}tan(30°) \tag{3.1}$$

3.1.2 For staggered bolts

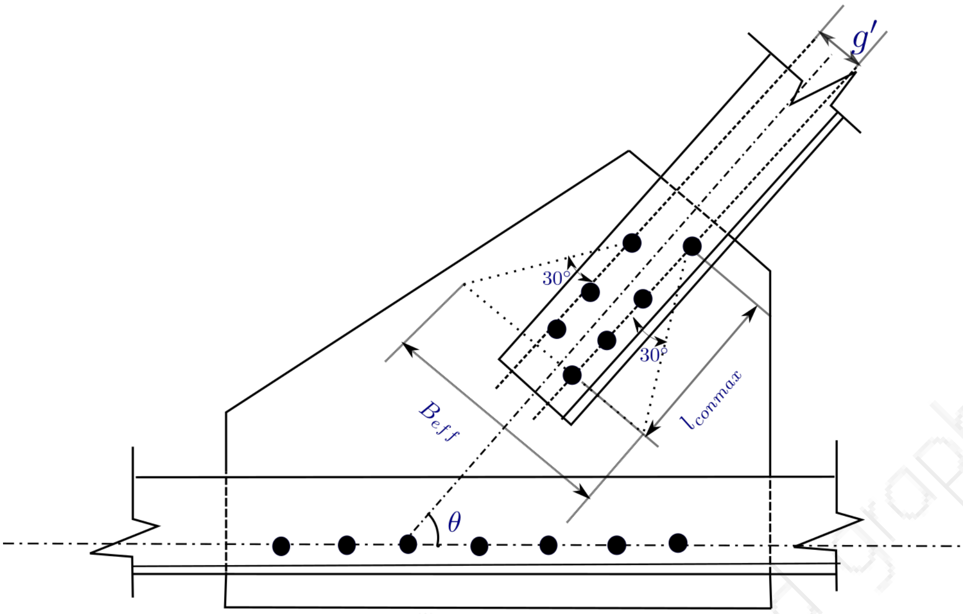


Figure 3.2: Whitmore Effective Width for Staggered bolting arrangement

$$B_{eff} = g' + (2l_{conmax}tan(30°)) \quad (3.2)$$

where,
g' = gauge length between the staggered bolts;
l_{conmax} = maximum connection length of staggered bolts for each member.

3.2 Check for Tension

3.2.1 Check for Tension Yielding

[Reference: Cl 6.2 IS 800: 2007]

$$\frac{P}{(B_{eff}t_g)} \leq \frac{f_y}{\gamma_{m0}} \quad (3.3)$$

where,
P = Factored Axial force on each member;
B_{eff} = Whitmore effective width;
t_g = Gusset plate thickness;
f_y = yield strength of plate;
γ_{m0} = Partial factor of safety for yielding.

3.2.2 Check for Tension Rapture

[Reference: Cl 6.3 IS 800: 2007]

$$\frac{P}{(B_{eff} - nd_0)t_g} \leq \frac{0.9f_u}{\gamma_{m1}} \quad (3.4)$$

where,
P = Axial force on each member;
d_0 = bolt hole diameter;
n = numbers of bolts
B_{eff} = Whitmore effective width;
t_g = Gusset plate thickness;
f_u = ultimate strength of plate;
γ_{m0} = Partial factor of safety.

3.3 Whitmore equivalent column length (l_c)

[Reference: Page No. 365 Design of steel by structure N.Subramanian 13th Edition]

Whitmore equivalent column length is defined as average of the three lengths projected from the whitmore section,section, in the direction of member, to the fastener lines of the adjoining members. The three lengths are taken at the two ends of the width and at the center. If Whitmore section intersects an adjoining member bolt line, the length at that end is assumed to be zero. Conservatively, Whitmore equivalent column length can also be taken as the distance from center of member to the fasteners lines of adjoining members.

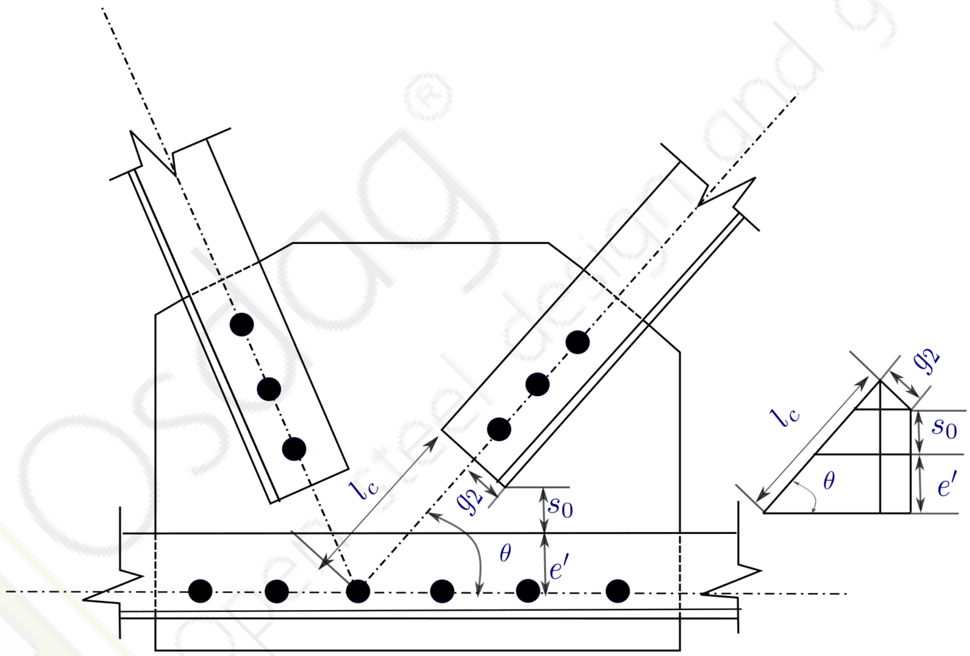


Figure 3.3: Equivalent Column Length

$$l_c = \frac{s_0 + g_1 + g_2 cos(\theta)}{sin(\theta)} \tag{3.5}$$

where,
s_0 = spacing between bottom chord and the connected truss member;
e'= edge distance of bottom chord;
g_2= gauge length of connected truss member;
θ = angle of inclination of connected truss member with bottom bottom chord.

3.4 Check for Buckling Failure

[Reference:Cl 7.1.2.1 IS 800: 2007]

$$f_{cd} = \frac{f_y/\gamma_{m0}}{\phi + \sqrt{\phi^2 - \lambda^2}} \tag{3.6}$$

where,
ϕ = $0.5[1 + \alpha(\lambda - 0.2) + \lambda^2]$
λ = non-dimensional effective slenderness ratio;
=$\sqrt{f_y/f_{cc}}$
f_{cc} = Euler Buckling stress=$\frac{\pi^2 E}{(Kl_c/r_{min})^2}$
K = 0.7,Effective length factor;
l_c = Whitmore equivalent column length;
r_{min}= radius of gyration of plate= =$t_g/\sqrt{12}$;
f_{cd} = Design Compressive stress;
t_g = Thickness of gusset plate.

$$\frac{F}{B_{eff}t_g} \leq f_{cd} \tag{3.7}$$

where,
F = Factored Axial force on truss member;
B_{eff}= Whitmore effective width.

Check 4

Check for Truss Member

4.1 Block failure check

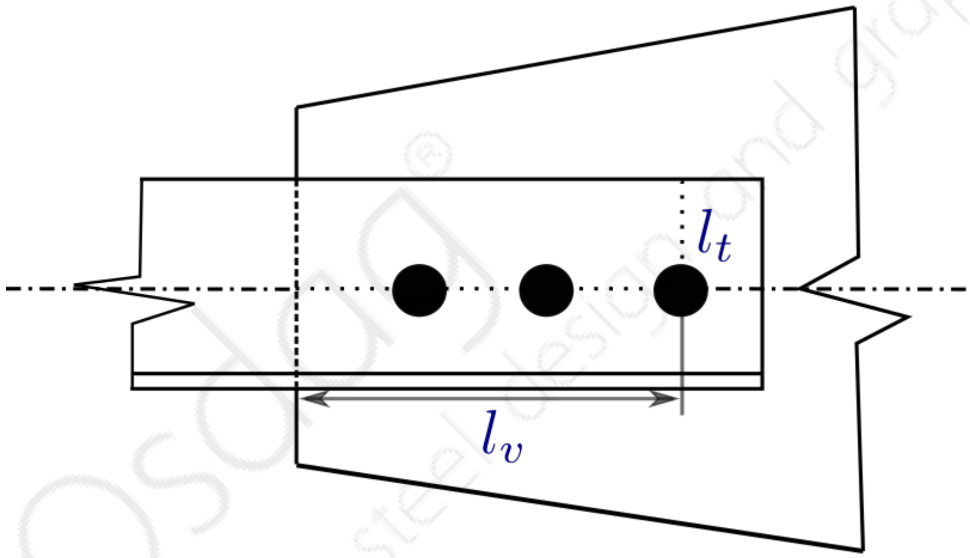


Figure 4.1: Block Failure of Truss member

$$T_{db1} = \frac{A_{vg} f_y}{\sqrt{3}\gamma_{m0}} + \frac{0.9 A_{tn} f_u}{\gamma_{m1}} \tag{4.1}$$

$$T_{db2} = \frac{0.9 A_{vn} f_u}{\sqrt{3}\gamma_{m1}} + \frac{A_{tg} f_y}{\gamma_{m0}} \tag{4.2}$$

where,
A_{vg} = Minimum gross area in shear along bolt line parallel to external force = $l_v \, * \, t$
A_{vn} = Minimum net area in shear along bolt line parallel to external force = $A_{vg} - (n_r - 0.5)d_0$
A_{tg} = Minimum gross area in tension from the bolt hole to the toe of the angle, end bolt line, perpendicular to the line of force=$l_t * t$
A_{tn} = Minimum net area in tension from the bolt hole to the toe of the angle, end bolt line, perpendicular to the line of force=$A_{tg} - (n_c - 0.5)d_0$
n_r = number of bolt rows
n_c = number of bolt columns
f_u = Ultimate stress of the plate material
f_y = Yield stress of the plate material
γ_{m0} = 1.10
γ_{m1} = 1.25

Check 5

Detailing

The size and shape of the gusset plates are decided based on the direction of various members meeting at a joint. The plate outlines are fixed so as to meet the minimum edge distance specified for the bolts that are used to connect various members at a particular joint and also it should give aesthetic appearance.

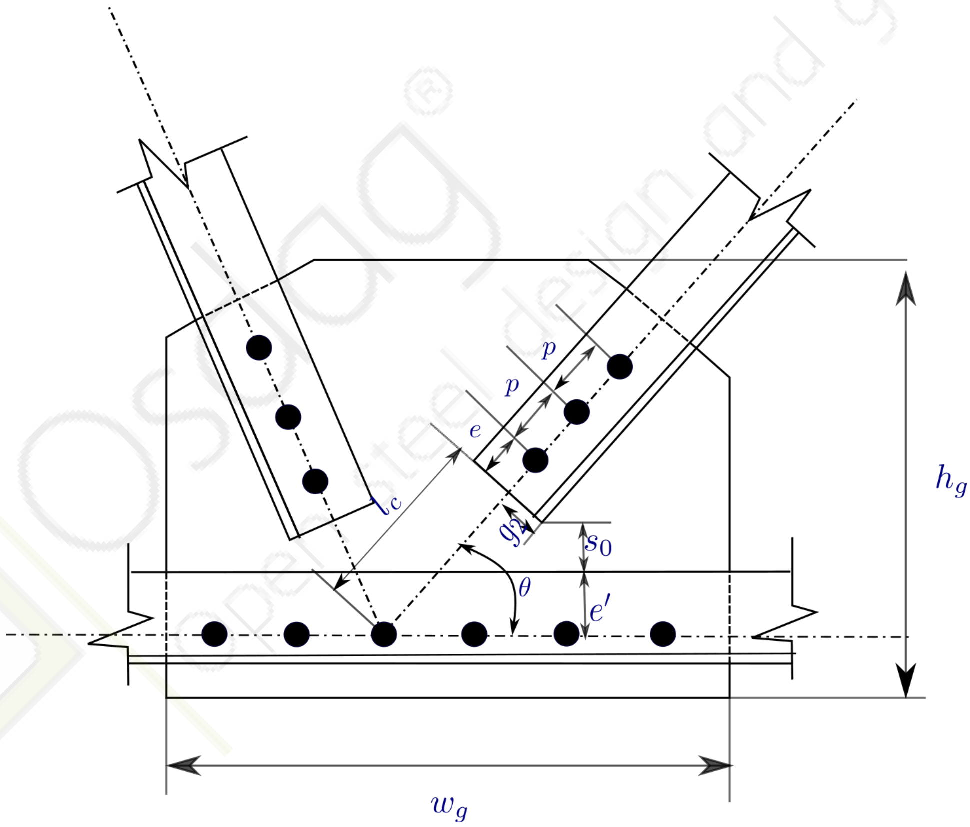


Figure 5.1: Gusset Plate detailing

5.1 Pitch (p) and Gauge (g)

[Reference: Cl. 10.2.2 and 10.2.3, IS 800 : 2007]

$$2.5\,d \;\leq\; p \; or \; g \;\leq\; \min(32\,t,\; 300\text{ mm}) \tag{5.1}$$

Where,
d = diameter of bolt;
t = thickness of the thinner plate.

5.2 End (e) and Edge (e')

[Reference: Clause 10.2.4, IS 800 : 2007]

$$[1.5 \text{ or } 1.7] \times d_0 \;\leq\; e/e' \;\leq\; 12\, t\, \varepsilon \tag{5.2}$$

$$\varepsilon = \sqrt{\frac{250}{f_y}} \tag{5.3}$$

Where,
1.5 for machine cut and 1.7 for hand cut;
d_0 = diameter of the hole;
t = thickness of the thinner plate;
f_y = yield stress of the plate.

5.3 Gusset plate Dimensions

5.3.1 Height of Gusset plate (h_g)

$$h_g = max[(l_c + l_{con} + 2e)sin(\theta)] \tag{5.4}$$

5.3.2 Width of gusset plate (w_g)

5.3.2.1 case 1

When truss connection has no member in θ_{max} < 90° or $\theta_{min} > 90°$, width of gusset plate is given by;

$$L_1 = max[(l_{con} + l_c + 2e)cos(\theta)] \tag{5.5}$$

$$L_2 = p(n-1) + e \tag{5.6}$$

$$w_g = L_1 + L_2 \tag{5.7}$$

5.3.2.2 case 2

When truss connection has members in θ < 90° and $\theta > 90°$,width of guesst plate is given by;

5.3.2.3 For ($\theta < 90°$)

$$L_1 = max[(l_{con_i} + l_{c_i} + 2e)cos(\theta)] \tag{5.8}$$

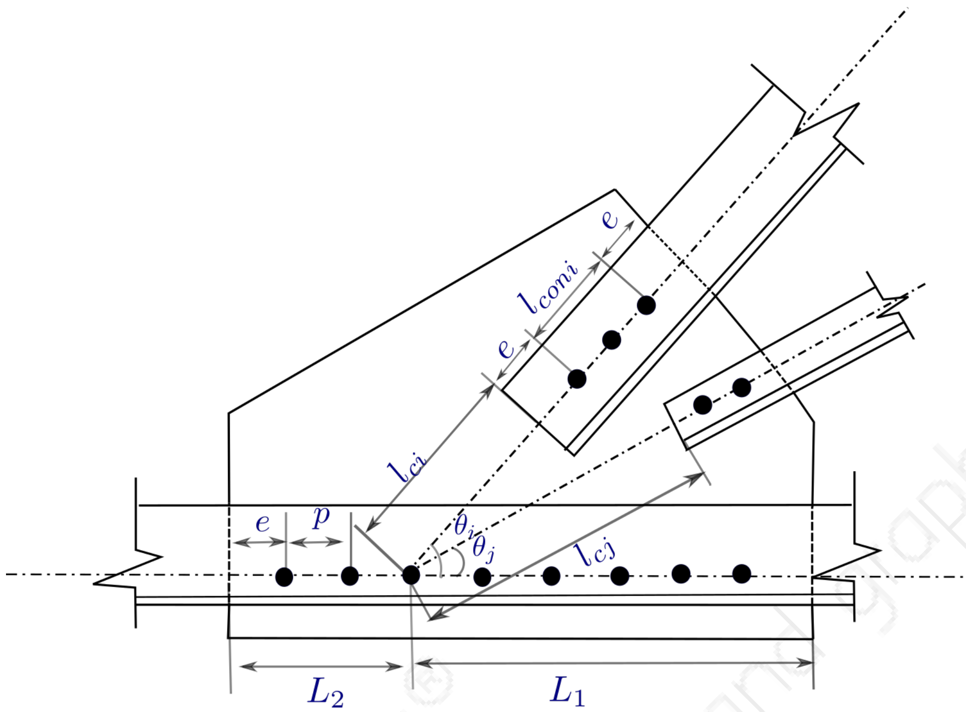


Figure 5.2: Width of Gusset Plate

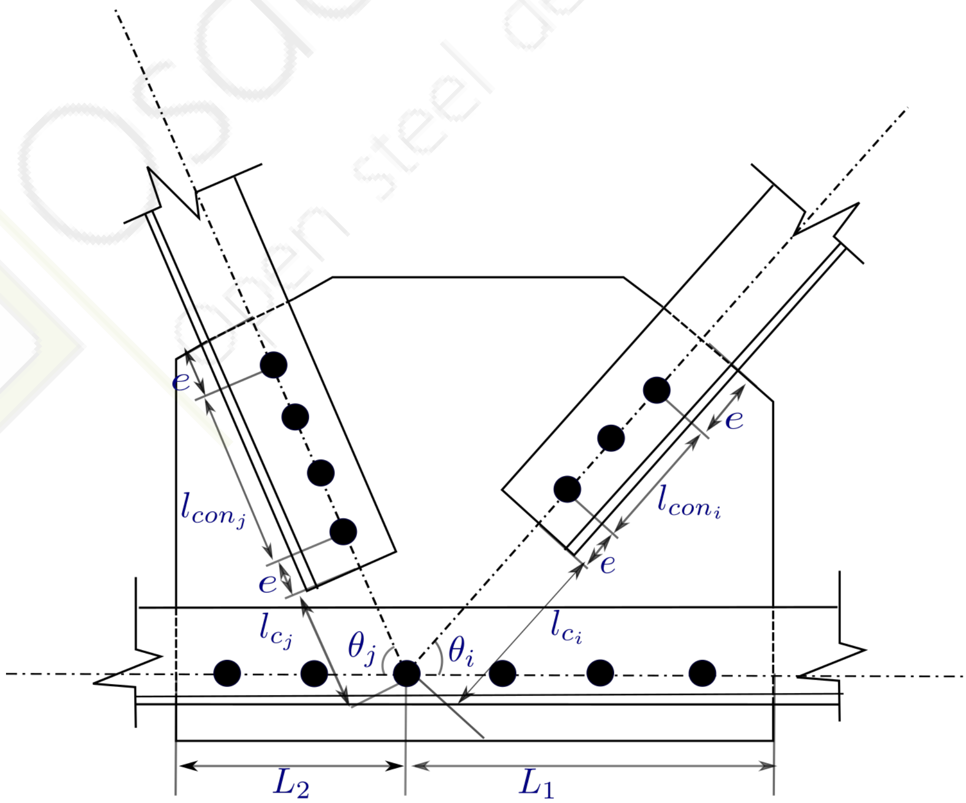


Figure 5.3: Width of Gusset Plate

5.3.2.4 For ($\theta > 90°$)

$$L_2 = max(l_{con_j} + l_{c_j} + 2e)cos(\theta)) \tag{5.9}$$

$$w_g = L_1 + L_2 \tag{5.10}$$

where,
n = number of bolts in bottom chord.
l_{con} = connection length of truss members;
l_c = Whitmore equivalent column length;
e = end distance for truss member;
p = pitch for the truss member;
n = number of bolts;
θ_{max} = angle of inclination of diagonal or vertical truss member with C.G of bottom chord.

Appendix G

DDCL for Welded Truss Connection

Osdag®
Open steel design and graphics

Design and Detailing Checklist (DDCL)
and
Design and Detailing Query (DDQ)

Welded Truss Connection

Prepared by:
Rachna Gupta

Under the guidance of:
Prof. Siddhartha Ghosh

Indian Institute of Technology, Bombay

July 13, 2019

Contents

Reviewer Details & Guideline(s) for filling DDCL/DDQ

1. Name of the reviewer:

2. Institute/Company/Organization:

3. Designation:

- This document is for Design and Detailing Check List (DDCL) and Design and Detailing Query (DDQ) created by Osdag team, IIT Bombay.
- The checks are documented algorithmically and chapter wise in the document where the checks and sub-checks are given in the section and subsection respectively.
- Reference of the check is given just below the check title

- Each check has an associated checkbox and a comment box for giving feedback. The checkbox can be checke
- If you check on the 'Not Ok' checkbox during the review, please specify your reason in the comment box with reference(s) (if any). It is mandatory!
- Send the document after review to sghosh@civil.iitb.ac.in.

  For any queries or help on filling the feedback document, contact Rachna Gupta at [rachna291998@gmail.com]

Reviewer Details

User Inputs

- Connecting members
  - Truss Member Sections*
- Connectivity*
  - Welded
- Material Property
  - f_u (MPa)*
  - f_y (MPa)*
- Factored loads
  - Axial Force (kN)
- Detailing
  - Angle of inclination(degree)*
- Plate
  - Thickness (mm)*
  - f_u (MPa)*
  - f_y (MPa)*
- Weld size
  - weld size (mm)*

Design and Detailing Checks

Check 1

Weld Design

Truss members and gusset plates are connected by weld

1.1 Gusset plate thickness (t_g)

Thickness of the gusset plate provided is usually equal to or slightly higher than members that are connected to gusset plate.
Thickness of gusset plate is calculated on the basis of maximum force acting on the truss members and is given by the table. [Reference:Table 5.12 Design of steel structure by N.Subramanian]

Maximum design force in diagonal(KN)	upto 200	200-450	450-750	750-1650	1650-2250	2250-3000
Thickness of gusset plate(mm)	8	10	12	16	18	20

1.2 Weld Length of Truss Members(l_w)

[Reference: Page no. 484 Design of steel structure by N.Subramanian Edition 13th]

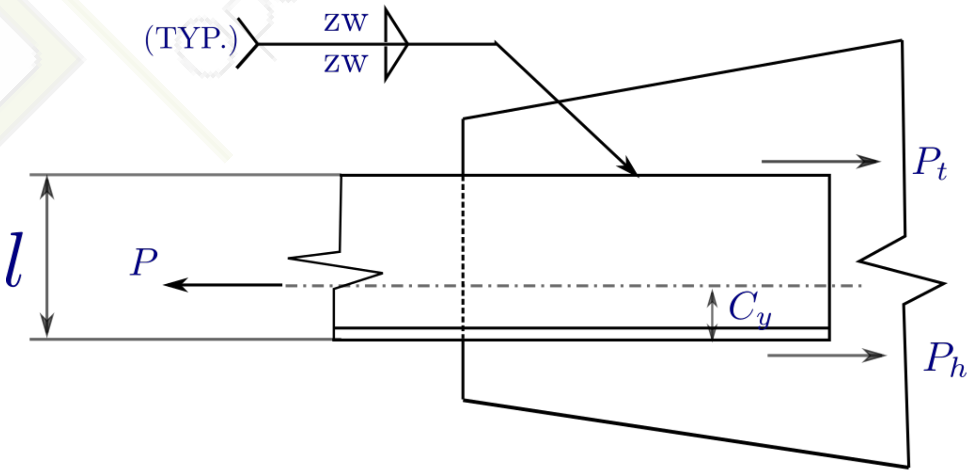


Figure 1.1: Balancing the weld on a tension member connection

$$f_w = \frac{t_t f_u}{\sqrt{3}\gamma_{m1}} \tag{1.1}$$

$$P_t = \frac{F C_y}{l} \tag{1.2}$$

$$P_h = F - P_t \tag{1.3}$$

where,
f_w = Strength of weld;
F= Axial Force on truss member;
P_h = Axial force resisted by weld in hill side;
P_t = Axial force resisted by weld in toe side;
C_y = C.G of bottom chord;
t_t= Throat thickness of weld = 0.7s;
s = weld size;
f_u= minimum of ultimate strength of truss member and welded material;
γ_{m1} = Partial safety factor for ultimate strength;
1 = leg size of truss member section.

$$l_{wt} = \frac{P_t}{f_w} \tag{1.4}$$

$$l_{wh} = \frac{P_h}{f_w} \tag{1.5}$$

where,
l_{wt} = Overall length of weld in toe side;
l_{wh} = Overall length of weld in hill side;

$$l_{wteff} = l_{wt} - 2s \tag{1.6}$$

$$l_{wheff} = l_{wh} - 2s \tag{1.7}$$

where,
l_{wteff} = Effective length of weld in toe side;
l_{wheff} = Effective length of weld in hill side.

Check 2

Gusset Plate Checks

2.1 Whitmore Effective Width (B_{eff})

[Reference: 5.12 Truss Joint Connection Page no. 365 Design of steel structure by N.Subramanian Edition 13th]

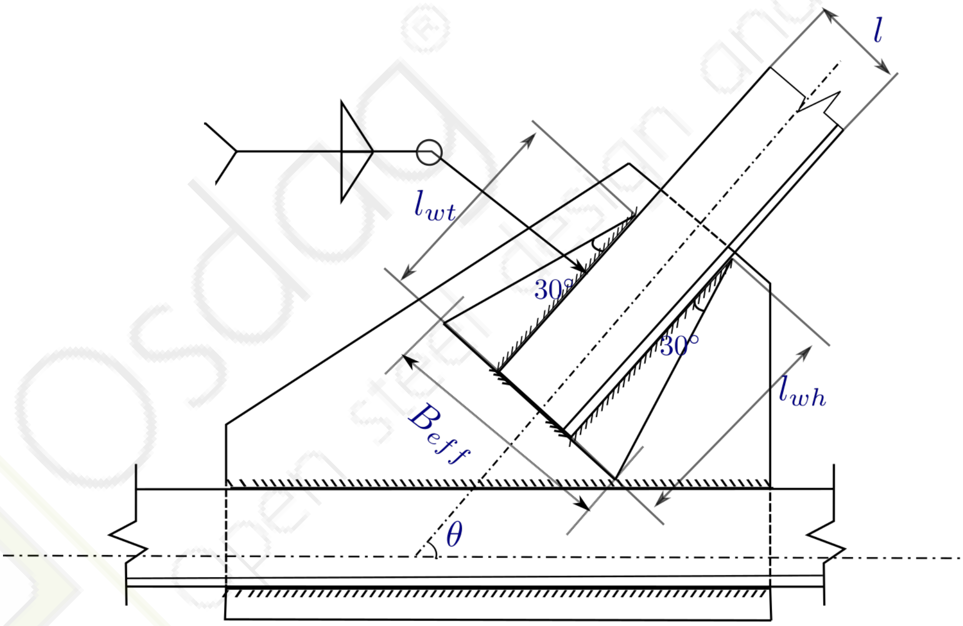


Figure 2.1: Effective Width of Gusset Plate as per Whitmore

$$B_{eff} = l + l_{wt}tan(30°) + l_{wh}tan(30°) \tag{2.1}$$

2.2 Check for Tension

2.2.1 Check for Tension Yielding

[Reference: Cl 6.2 IS 800: 2007]

$$\frac{P}{(B_{eff}t_g)} \leq \frac{f_y}{\gamma_{m0}} \tag{2.2}$$

where,
P = Axial force on each member;

B_{eff} = Whitmore effective width;
t_g = Gusset plate thickness;
f_y = yield strength of plate;
γ_{m0} = Partial factor of safety for yielding.

2.3 Compression Check

2.3.1 Whitmore equivalent column length (l_c)

[Reference:Design of steel by structure N.Subramanian]

$$l_c = \frac{s_0 + g_1 + g_2 cos(\theta)}{sin(\theta)} \tag{2.3}$$

where,
s_0 = spacing between bottom chord and the connected truss member;
g_1= gauge length of bottom chord;
g_2= gauge length of connected truss member;
θ = angle of inclination of connected truss member with bottom bottom chord.

2.4 Check for Buckling Failure

[Reference:Cl 7.1.2.1 IS 800: 2007]

$$f_{cd} = \frac{f_y/\gamma_{m0}}{\phi + \sqrt{\phi^2 - \lambda^2}} \tag{2.4}$$

where,
ϕ = $0.5[1 + \alpha(\lambda - 0.2) + \lambda^2]$
λ = non-dimensional effective slenderness ratio;
=$\sqrt{f_y/f_{cc}}$
f_{cc} = Euler Buckling stress=$\frac{\pi^2 E}{(Kl_c/r_{min})^2}$
K = 0.7,Effective length factor;
l_c = Whitmore equivalent column length;
r_{min}= radius of gyration of plate= =$t_g/\sqrt{12}$;

f_{cd} = Design Compressive stress;
t_g = Thickness of gusset plate.

$$\frac{F}{B_{eff} t_g} \leq f_{cd} \tag{2.5}$$

where,
F = Axial force on truss member;
B_{eff}= Whitmore effective width.

2.5 Block Failure of Gusset Plate

[Reference:Cl 6.4.1 IS 800: 2007]

In case of welded connection gusset plate may fail by block shear may occur.

$$T_{db1} = \frac{A_{vg} f_y}{\sqrt{3}\gamma_{m0}} + \frac{0.9 A_{tn} f_u}{\gamma_{m1}} \tag{2.6}$$

$$T_{db2} = \frac{0.9 A_{vn} f_u}{\sqrt{3}\gamma_{m1}} + \frac{A_{tg} f_y}{\gamma_{m0}} \tag{2.7}$$

where,
A_{vg} = Minimum gross area in shear along bolt line parallel to external force = $l_v \, * \, t$
A_{vn} = Minimum net area in shear along bolt line parallel to external force = $l_v \, * \, t$
A_{tg} = Minimum gross area in tension from the bolt hole to the toe of the angle, end bolt line, perpendicular to the line of force=$l_t * t$
A_{tn} = Minimum net area in tension from the bolt hole to the toe of the angle, end bolt line, perpendicular to the line of force=$l_t * t$
f_u = Ultimate stress of the plate material
f_y = Yield stress of the plate material
γ_{m0} = 1.10
γ_{m1} = 1.25

Check 3

Detailing

3.1 Gusset plate Dimensions

3.1.1 Height of Gusset plate (h_g)

Height of gusset plate is based on maximum weld length and angle of inclination of truss member with C.G of bottom chord.

$$h_g = max(lc + l_w)sin(\theta) \tag{3.1}$$

where,
l_w = weld length of diagonal or vertical truss member;
l_c = Whitmore equivalent column length;
θ angle of inclination of truss member with C.G of bottom chord.

3.1.2 Width of gusset plate (w_g)

Width of gusset plate is equal to weld length of bottom chord.

$$w_g = L_{wb} \tag{3.2}$$

where,
L_{wb} =Length of weld for bottom chord.