FOSSEE Winter Internship Report

On

Development of GUI Components and Welded Connection Calculations for Osdag

Submitted by

Tanu Singh

3rd Year B.Tech Student, Department of Computer Science and Enginnering

Kalinga Institute of Industrial Technology

Bhubaneswar

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

July 18, 2025

Acknowledgments

- I would like to express my sincere gratitude to all those who supported and guided me throughout my internship with Osdag. This experience has been invaluable for my professional development in structural engineering software.
- My deepest appreciation goes to the entire Osdag team for their mentorship, particularly to Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia for their patient guidance and technical expertise during my project work.
- I am honored to acknowledge the leadership of Prof. Siddhartha Ghosh, Principal Investigator of Osdag from the Department of Civil Engineering at IIT Bombay, for creating this impactful open-source initiative.
- Special thanks to Prof. Kannan M. Moudgalya, FOSSEE Project Investigator from the Department of Chemical Engineering at IIT Bombay, for his vision in supporting open-source engineering education.
- I gratefully acknowledge the support from FOSSEE managers Usha Viswanathan and Vineeta Parmar, along with their entire team, for creating opportunities that bridge academia and practical software development.
- This project was made possible through the support of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India.
- I extend my thanks to Kalinga Institute of Industrial Technology for fostering an environment that encourages practical learning through such internship opportunities.

Contents

1 Introduction **4**

1.1 National Mission in Education through ICT 4

1.1.1 ICT Initiatives of MoE 5

1.2 FOSSEE Project 6

1.2.1 Projects and Activities 6

1.2.2 Fellowships 6

1.3 Osdag Software 7

1.3.1 Osdag GUI 8

1.3.2 Features 8

2 Screening Task **9**

2.1 Problem Statement 9

2.2 Tasks Done 9

3 Butt Joint Welded Module Development **10**

3.1 Task 1: Problem Statement 10

3.2 Task 1: Tasks Done 10

3.2.1 Graphical User Interface Development 10

4 Internship Task 2: Butt Joint Welded Connection Calculations **12**

4.1 Task 2: Problem Statement 12

4.2 Task 2: Tasks Done 12

4.2.1 Mathematical Formulations Implementation 12

4.3 Task 2: Documentation 13

4.3.1 Core Calculation Methods 13

5 Lap Joint Welded Module Development **22**

5.1 Task 1: Problem Statement 22

5.2 Task 1: Tasks Done 22

5.2.1 Graphical User Interface Development 22

6 Internship Task 4: Lap Joint Welded Connection Calculations **24**

6.1 Task 2: Problem Statement 24
6.2 Task 2: Tasks Done 24
6.2.1 Mathematical Formulations Implementation 24
6.3 Task 2: Documentation 25
6.3.1 Weld Calculation Methods 25

7 Conclusions **32**
7.1 Tasks Accomplished 32
7.1.1 GUI Development for Butt Joint Welded 32
7.1.2 Calculation Implementation for Butt Joint Welded Connection . . 33
7.1.3 Lap Joint Welded 33
7.2 Skills Developed 34
7.2.1 Technical Skills 34

Bibliography **39**

Chapter 1

Introduction

1.1 National Mission in Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: www.nmeict.ac.in.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion**: Porting solved examples from textbooks using FLOSS.
- **Lab Migration**: Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities**: Specialized activities to promote niche software tools.
- **Forums**: Providing a collaborative space for users.
- **Workshops and Conferences**: Organizing events to train and inform users.

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices [1, 2, 3]. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of

- **Input Dock**: Collects and validates user inputs.
- **Output Dock**: Displays design results after validation.
- **CAD Window**: Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log**: Shows errors, warnings, and suggestions based on design checks.

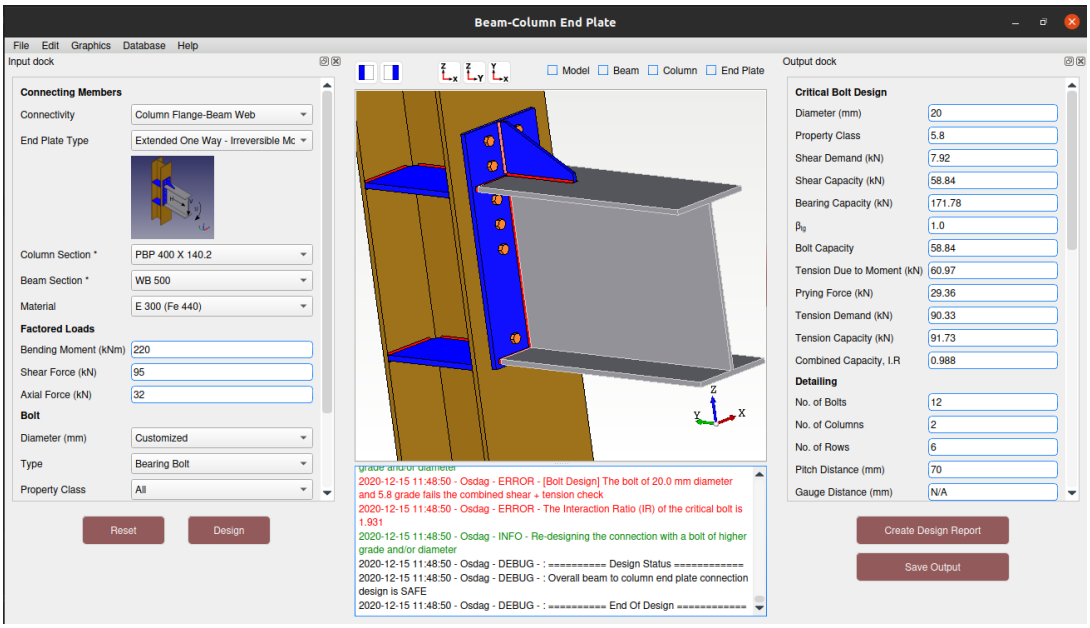

Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model**: The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences**: Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report**: Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Screening Task

2.1 Problem Statement

State the problem defined during the screening task.

2.2 Tasks Done

List and describe the tasks performed as part of the screening task.

Chapter 3

Butt Joint Welded Module Development

3.1 Task 1: Problem Statement

Developed a comprehensive Butt Joint Welded design interface for Osdag software to enable structural engineers to:

- Input all necessary design parameters through an intuitive interface
- Visualize plate girder configurations in 3D
- Generate IS 800:2007 compliant designs
- Automate calculation of cover plate properties and weld requirements

3.2 Task 1: Tasks Done

3.2.1 Graphical User Interface Development

- Designed a tabbed interface with logical parameter grouping:
  - General Parameters (Material, Structure Type, Restraint Conditions)
  - Geometric Dimensions
  - Loading Conditions

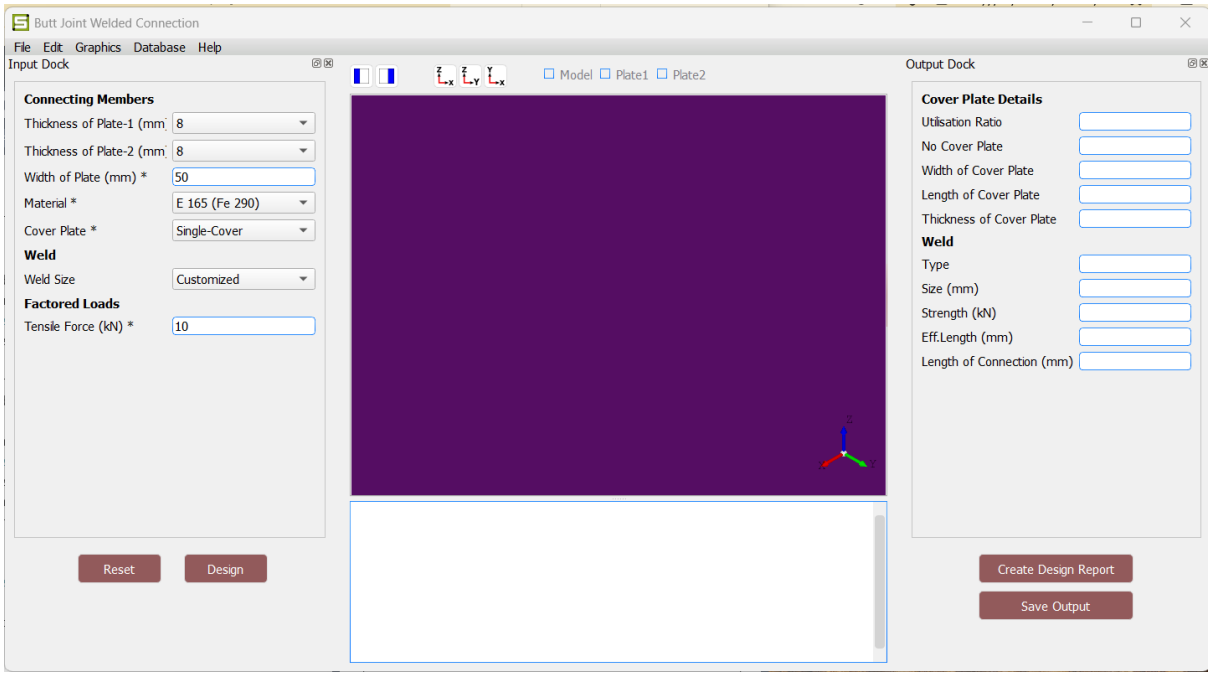

Figure 3.1: GUI of Butt Joint Welded

- Implemented dynamic input validation for:
  - Material grades (E250 to E450)
  - Standard thickness values (8mm to 63mm) length limits (1m to 30m)
- Created interactive selection modes:
  - Optimized (software-calculated values)
  - Custom (user-specified values)

Chapter 4

Internship Task 2: Butt Joint Welded Connection Calculations

4.1 Task 2: Problem Statement

The objective was to implement the calculation engine for welded butt joint connections according to IS 800:2007. This involved developing mathematical formulations and algorithms for:

- Cover plate thickness calculations
- Weld capacity calculations
- Packing plate thickness
- Weld spacing and arrangement calculations
- Design adequacy checks and utilization ratios

4.2 Task 2: Tasks Done

4.2.1 Mathematical Formulations Implementation

- Implemented cover plate thickness calculations as per IS 800:2007
- Developed weld capacity calculation algorithms

- Created reduction factor calculations
- Implemented spacing and detailing calculations
- Added packing plate thickness calculations

4.3 Task 2: Documentation

4.3.1 Core Calculation Methods

Cover Plate Thickness Calculations

The cover plate thickness is calculated based on the connection type and minimum plate thickness as per IS 800:2007.

For Single Cover Plate:

$$T_{cp} = \frac{5}{8} \times T_{min}$$

For Double Cover Plate:

$$T_{cp} = \frac{9}{8} \times T_{min}$$

- T_{cp} = Cover plate thickness (mm)
- T_{min} = Minimum thickness of main plates (mm)

Implementation:

Listing 4.1: Butt Joint Welded Connection Design in Osdag

```
%--------------------begin code-------------
plate1_thk = float(design_dictionary[KEY_PLATE1_THICKNESS])
plate2_thk = float(design_dictionary[KEY_PLATE2_THICKNESS])
Tmin = min(plate1_thk, plate2_thk)
cover_plate_type_str = design_dictionary[KEY_COVER_PLATE]

available_thicknesses = [float(thk) for thk in PLATE_THICKNESS_SAIL]
if "double" in cover_plate_type_str.lower():
    self.planes = 2
```

```
    Tcp = math.ceil((9.0 / 16.0) * Tmin)  # Double cover plate
        thickness as per Eq. 3.2
    self.calculated_cover_plate_thickness = min([thk for thk in
        available_thicknesses if thk >= Tcp],default=Tcp)

    # Packing plate logic as per Cl. 10.3.3.2
    if abs(plate1_thk - plate2_thk) > 0.001:
        self.packing_plate_thickness = abs(plate1_thk - plate2_thk)
    else:
        self.packing_plate_thickness = 0.0

elif "single" in cover_plate_type_str.lower():
    self.planes = 1
    Tcp = math.ceil((5.0 / 8.0) * Tmin)  # Single cover plate thickness
        as per Eq. 3.1
    self.calculated_cover_plate_thickness = min(
                [thk for thk in available_thicknesses if thk >= Tcp],
                default=Tcp
            )
    self.packing_plate_thickness = 0.0

else:
    self.planes = 1
    Tcp = Tmin
    self.calculated_cover_plate_thickness = min(
                [thk for thk in available_thicknesses if thk >= Tcp],
                default=Tcp
            )
    self.packing_plate_thickness = 0.0

%-------------------- end code ---------------
```

Packing Plate Calculations

When there's a thickness difference between plates, packing plates are required.

$$t_{pkg} = |t_1 - t_2| \tag{4.3}$$

Implementation:

```
%--------------------begin code-------------
if abs(plate1_thk - plate2_thk) > 0.001:
                self.packing_plate_thickness = abs(plate1_thk -
                   plate2_thk)
            else:
                self.packing_plate_thickness = 0.0
%-------------------- end code ---------------
```

Design of Weld

For fillet welds connecting cover plates, the effective throat thickness and design strength of weld are computed as follows:

1. **Effective throat:** $a = 0.707 \times s$
2. **Design strength of weld:**
$$f_{wd} = \frac{f_y}{\sqrt{3} \times \gamma_{mw}}$$
3. **Limit weld size:**
   - s_{min}: As per Table 21 of IS 800:2007
   - $s_{max} = T_{min} - 1.5\,\text{mm}$

Implementation:

```
%--------------------begin code-------------
self.effective_throat_thickness = 0.707 * self.weld_size
        self.fu = float(design_dictionary[KEY_DP_WELD_MATERIAL_G_O])
        weld_type = design_dictionary[KEY_DP_WELD_TYPE]
        if weld_type == "Shop weld":
            self.gamma_mw = 1.25
        else:
            self.gamma_mw = 1.50
        self.weld_design_strength = self.fu / (math.sqrt(3) * self.
           gamma_mw)
%-------------------- end code ---------------
```

Required Weld Length

The required length of fillet weld is calculated to resist the applied tensile force.

1. Convert: $P_N = P \times 10^3$
2. Required weld length:
$$L_{\text{req}} = \frac{P_N}{f_{wd} \times a \times n_f}$$
3. If $L_{\text{req}} \leq w$: provide straight welds
4. If $L_{\text{req}} > w$: proceed to Section 3.4

Implementation:

```
%--------------------begin code-------------
if self.L_req <= self.plates_width:
            logger.info(": Straight weld will be provided as required
                length is less than plate width")
            self.weld_length_provided = self.plates_width
            self.weld_length_effective = self.weld_length_provided
            self.weld_angle = 0
            self.side_weld_length = 0

else:
            # Calculate skewed weld parameters
            L_target = self.L_req / self.N_f  # Required length per
                weld line

%-------------------- end code ---------------
```

Weld-Length Extension

When the required weld length exceeds the plate width, skewed ends or side welds are used to accommodate additional weld length.

1. $L_{\text{tgt}} = \frac{L_{\text{req}}}{n_f}$
2. $L_{\text{line}} = w + 2w \times \tan(\alpha)$

3. $\alpha = \arctan\left(\frac{L_{\text{tgt}} - w}{2w}\right)$

4. If $\alpha < 20°$, set $\alpha = 20°$; if $\alpha > 60°$, set $\alpha = 60°$ and proceed to Section 3.4.3

Implementation:

```
%--------------------begin code-------------
L_target = self.L_req / self.N_f  # Required length per weld line

            # Calculate skew angle
            self.weld_angle = math.degrees(math.atan((L_target - self.
                plates_width)/(2 * self.plates_width)))

            # Constrain angle between 20-60 degrees
            if self.weld_angle < 20:
                self.weld_angle = 20
            elif self.weld_angle > 60:
                self.weld_angle = 60

%-------------------- end code ---------------
```

Weld Strength Verification

When the required weld length exceeds the plate width, skewed ends or side welds are used to accommodate additional weld length.

1. $L_{\text{eff}} = L_{\text{provided\_line}} - 2a$

2. Ensure $L_{\text{eff}} \geq 4s$

3. $C_w = f_{wd} \times a \times L_{\text{eff}} \times n_f$

4. Ensure $P_N \leq C_w$

Implementation:

```
%--------------------begin code-------------
        min_length = 4 * self.weld_size
        if self.weld_length_effective < min_length:
            self.design_status = False
```

```
            logger.error(": Effective weld length {} mm is less than minimum required length {} mm [Ref. Cl.10.5.4, IS 800:2007]".format(
                round(self.weld_length_effective, 2), round(min_length, 2)))
            logger.info(": Increase the weld length or size")
            return
        else:
            logger.info(": Minimum length requirement satisfied")

        # Calculate weld strength
        self.weld_strength = self.f_w * 0.707 * self.weld_size * self.weld_length_effective * self.N_f

        # Calculate weld utilization ratio
        weld_utilization = self.tensile_force / self.weld_strength
        self.utilization_ratios['weld'] = weld_utilization

%-------------------- end code ---------------
```

Reduction Factor for Long Joints

When the required weld length exceeds the plate width, skewed ends or side welds are used to accommodate additional weld length.

1. Check if $L_{eff} > 150a$
2. If so, compute:
$$\beta_L = 1.2 - \frac{0.2L_{eff}}{150a}$$
3. Ensure $\beta_L \geq 0.8$
4. Adjust the design strength of weld:
$$f_{wd}^{adj} = \beta_L \times f_{wd}$$
5. Use f_{wd}^{adj} instead of f_{wd} in weld strength calculations.

Implementation:

```
%--------------------begin code-------------
        a = 0.707 * self.weld_size

        # Check if reduction is needed
        if self.weld_length_effective <= 150 * a:
            self.beta_L = 1.0
            logger.info(": No reduction for long joints required as
               length is less than 150 times throat thickness")
            return

        # Calculate reduction factor
        self.beta_L = 1.2 - (0.2 * self.weld_length_effective)/(150 * a
           )

        # Ensure minimum value of 0.8
        self.beta_L = max(0.8, self.beta_L)

        # Adjust weld design strength and recalculate utilization
        self.f_w_adjusted = self.f_w * self.beta_L

        # Recalculate weld strength with reduction factor
        self.weld_strength_reduced = self.f_w_adjusted * 0.707 * self.
           weld_size * self.weld_length_effective * self.N_f

%-------------------- end code ---------------
```

Base Metal Strength Check

When the required weld length exceeds the plate width, skewed ends or side welds are used to accommodate additional weld length.

1. Check if $L_{eff} > 150a$
2. If so, compute:
$$\beta_L = 1.2 - \frac{0.2 L_{eff}}{150a}$$
3. Ensure $\beta_L \geq 0.8$

4. Adjust the design strength of weld:

$$f_{wd}^{adj} = \beta_L \times f_{wd}$$

5. Use f_{wd}^{adj} instead of f_{wd} in weld strength calculations.

Implementation:

```
%--------------------begin code-------------

        Tmin = min(float(design_dictionary[KEY_PLATE1_THICKNESS]),
                   float(design_dictionary[KEY_PLATE2_THICKNESS]))
        self.A_g = Tmin * self.plates_width
        self.A_n = self.A_g  # For welded joints, net area equals gross
            area

        # Calculate design strength based on yielding and rupture
        T_dy = self.A_g * self.fy / self.gamma_m0
        T_du = 0.9 * self.A_n * self.fu / self.gamma_m1

        # Design base metal strength is minimum of the two
        self.T_db = min(T_dy, T_du)

        # Calculate base metal utilization ratio
        base_metal_utilization = self.tensile_force/self.T_db
        self.utilization_ratios['base_metal'] = base_metal_utilization


%-------------------- end code ---------------
```

Calculation Flow

The calculation process follows this sequence:

1. **Input Validation:** Validate all input parameters
2. **Cover Plate Design:** Calculate cover plate thickness
3. **Packing Plate Check:** Determine if packing plates are needed

4. **Bolt Selection:** Iterate through available bolt diameters and grades
5. **Capacity Calculation:** Calculate bolt capacities
6. **Number of Bolts:** Determine required number of bolts
7. **Spacing Design:** Calculate minimum and maximum spacing
8. **Bolt Arrangement:** Optimize bolt layout
9. **Capacity Reductions:** Apply long joint and large grip reductions
10. **Final Checks:** Verify design adequacy and calculate utilization ratio

Conclusion

The calculation engine successfully implements all required mathematical formulations for welded butt joint connections according to IS 800:2007. The implementation includes:

- Complete mathematical formulations with proper equations
- Robust calculation algorithms with error handling
- Iterative optimization for bolt selection
- Comprehensive capacity reduction factor calculations
- Spacing and arrangement optimization
- Design adequacy verification through utilization ratios

All calculations are implemented with proper validation, error handling, and compliance with the Indian Standard IS 800:2007, ensuring safe and efficient design of welded butt joint connections.

Chapter 5

Lap Joint Welded Module Development

5.1 Task 1: Problem Statement

Developed a comprehensive Lap Joint Welded design interface for Osdag software to enable structural engineers to:

- Input all necessary design parameters through an intuitive interface
- Visualize plate girder configurations in 3D
- Generate IS 800:2007 compliant designs
- Automate calculation of cover plate properties and weld requirements

5.2 Task 1: Tasks Done

5.2.1 Graphical User Interface Development

- Designed a tabbed interface with logical parameter grouping:
  - General Parameters (Material, Structure Type, Restraint Conditions)
  - Geometric Dimensions
  - Loading Conditions

- Implemented dynamic input validation for:
  - Material grades (E250 to E450)
  - Standard thickness values (8mm to 63mm) length limits (1m to 30m)
- Created interactive selection modes:
  - Optimized (software-calculated values)
  - Custom (user-specified values)

Chapter 6

Internship Task 4: Lap Joint Welded Connection Calculations

6.1 Task 2: Problem Statement

The objective was to implement the calculation engine for welded lap joint connections according to IS 800:2007. This involved developing mathematical formulations and algorithms for:

- Utilization ratio calculations
- Weld calculations

6.2 Task 2: Tasks Done

6.2.1 Mathematical Formulations Implementation

- Calculated Weld Strength
- Developed weld capacity calculation algorithms
- Created reduction factor calculations
- Implemented spacing and detailing calculations
- Added packing plate thickness calculations

6.3 Task 2: Documentation

6.3.1 Weld Calculation Methods

Weld Size Check Calculations

As per Cl. 10.5.2.3 of IS:800:2007 and Table 21, the minimum size of fillet weld depends on the thickness of the thicker part being joined.

- Thickness of thicker part up to 10 mm: Minimum weld size = 3 mm
- Thickness of thicker part over 10 mm up to 20 mm: Minimum weld size = 5 mm
- Thickness of thicker part over 20 mm up to 32 mm: Minimum weld size = 6 mm
- Thickness of thicker part over 32 mm up to 50 mm: Minimum weld size = 8 mm
- Thickness of thicker part over 50 mm: Minimum weld size = 10 mm

As per Cl. 10.5.2.4 of IS:800:2007, the maximum size of fillet weld is limited to prevent undercutting and ensure proper weld geometry:

For material less than 10 mm thick:

$$s_{\max} = t \tag{3.12}$$

For material 10 mm thick or more:

$$s_{\max} = t - 1.5\,\text{mm} \tag{3.13}$$

Where,

- t = Thickness of thinner part joined

Implementation:

```
%--------------------begin code-------------
s_min = IS800_2007.cl_10_5_2_3_min_weld_size(plate1_thk, plate2_thk)
        s_max = Tmin - 1.5 if Tmin >= 10 else Tmin
```

```
5
6           logger.info(f": Minimum weld size required (s_min) = {s_min} mm
                [Ref. Table 21, Cl.10.5.2.3]")
7           logger.info(f": Maximum allowed weld size (s_max) = {s_max} mm
               [Ref. Cl.10.5.3.1]")
8
9           selected_size = None
10          if isinstance(weld_size, str) and weld_size.lower() == 'all':
11              valid_sizes = [s for s in ALL_WELD_SIZES if s_min <= s <=
                    s_max]
12              if valid_sizes:
13                  selected_size = float(valid_sizes[0])
14          else:
15              try:
16                  size_val = float(weld_size[0] if isinstance(weld_size,
                        list) else weld_size)
17                  if s_min <= size_val <= s_max:
18                      selected_size = size_val
19              except (ValueError, IndexError):
20                  pass
21
22
23  %-------------------- end code ---------------
```

Required Weld Length Calculations

The required effective length of weld for a lap joint depends on the design strength of the weld per unit length and the factored tensile force to be transmitted.

Single-sided Welding For lap joints with weld on one side only:

$$l_{\text{req}} = \frac{P}{P_d} \tag{3.17}$$

Where,

- P = Factored tensile force
- P_d = Design strength of weld per unit length (from equation 3.5)
- l_{req} = Required effective length of weld

Double-sided Welding For lap joints with welds on both sides (most common configuration):

$$l_{\text{req}} = \frac{P}{2 \cdot P_d} \tag{3.18}$$

The factor 2 accounts for the two parallel welds sharing the load equally.

Implementation:

```
%--------------------begin code-------------
        self.weld_length_required = self.tensile_force / (2 * self.
            fillet_weld_design_strength)
        self.leff_min = max(4 * self.weld_size, 40)  # Cl.10.5.4.1
        self.leff_max = 70 * self.weld_size  # Cl.10.5.4.1

%-------------------- end code ---------------
```

Check Long Joint

Implementation:

```
%--------------------begin code-------------
if self.l_eff > 150 * self.effective_throat_thickness:
            self.beta_lw = 1.2 - 0.2 * (self.l_eff / (150 * self.
                effective_throat_thickness))
            self.beta_lw = max(0.6, min(self.beta_lw, 1.0))
            logger.info(f": Joint is long, reduction factor beta_lw = {
                self.beta_lw:.3f} [Cl.10.5.7.3]")
        else:
            logger.info(": No reduction for long joint required (Pass)"
                )
        # Modified required length
        l_req_modified = self.l_eff / self.beta_lw
        if l_req_modified < self.leff_min:
            logger.warning(f": Modified required weld length {
                l_req_modified:.2f} mm is less than minimum effective
                length {self.leff_min} mm [Cl.10.5.4.1]")
            self.l_eff = self.leff_min
        elif l_req_modified > self.leff_max:
            logger.error(": Modified required weld length exceeds
                maximum allowed. Increase weld size. [Cl.10.5.4.1]")
```

```
15              self.design_status = False
16              raise ValueError("Modified required weld length exceeds
                    maximum allowed.")
17          else:
18              self.l_eff = l_req_modified
19          # End return length (Cl.10.5.4.5): min(2*s, 12mm)
20          self.end_return_length = max(2 * self.weld_size, 12)  # Cl
                .10.5.4.5
21          logger.info(f": End return length = {self.end_return_length} mm
                [Cl.10.5.4.5]")
22          # Overlap length (Cl.10.5.4.3): min overlap = 4*s or 40mm,
                whichever is more
23          self.overlap_length = max(4 * self.weld_size, 40)
24          logger.info(f": Overlap length = {self.overlap_length} mm [Cl
                .10.5.4.3]")
25          self.connection_length = self.l_eff + 2 * self.
                end_return_length
26          # Design capacity (Cl.10.5.7.3):
27          self.design_capacity = 2 * self.l_eff * self.
                fillet_weld_design_strength * self.beta_lw
28          # Weld stress check (Cl.10.5.7):
29          self.weld_stress = self.tensile_force / (2 * self.
                effective_throat_thickness * self.l_eff) if self.l_eff else
                0
30
31  %-------------------- end code ---------------
```

Check Base Metal Strength

As per Cl. 10.5.7.1.2 of IS:800:2007, the design strength of the parent metal governs when the weld metal is stronger than the parent metal. The design strength of parent metal per unit length is given by:

$$P_{md} = 0.6 \cdot f_u \cdot \frac{t_l}{\gamma_{mw}} \tag{3.4}$$

Where,

- f_u: Ultimate tensile strength of parent metal
- t_l: Effective throat thickness

- γ_{mw}: Partial safety factor for weld = 1.25 (shop), 1.5 (field)

Implementation:

```
%--------------------begin code-------------
Tmin = min(float(design_dictionary[KEY_PLATE1_THICKNESS]), float(
    design_dictionary[KEY_PLATE2_THICKNESS]))
        self.A_g = Tmin * self.width
        self.gamma_m0 = 1.10
        self.gamma_m1 = 1.25
        # Shear lag factor (Cl.6.3.3):
        # For lap joints, net section efficiency = 0.7 (if not
            otherwise calculated)
        shear_lag_factor = 0.7
        T_dg = self.A_g * self.plate1.fy / self.gamma_m0  # Gross
            section yielding (Cl.6.2.2)
        T_dn = 0.9 * self.A_g * self.plate1.fu * shear_lag_factor /
            self.gamma_m1  # Net section rupture (Cl.6.2.3, 6.3.3)
        self.T_db = min(T_dg, T_dn)
        self.utilization_ratios['base_metal'] = self.tensile_force /
            self.T_db if self.T_db > 0 else float('inf')


%-------------------- end code ---------------
```

Calculation Flow for Welded Lap Joint

The calculation process for a welded lap joint in Osdag follows this general sequence:

1. **Input Validation:**
   - Verify plate thicknesses, widths, material grades, weld size, and factored tensile force.
   - Confirm design preferences such as edge preparation and fabrication type (shop/field).
2. **Weld Strength Calculation:**
   - Compute the effective throat thickness: $t_t = 0.7\,s$ for equal leg fillet welds.

- Calculate design strength of weld metal: $P_{wd} = \frac{f_u\, t_t}{\sqrt{3}\gamma_{mw}}$
- Calculate design strength of parent metal: $P_{md} = 0.6\, f_u\, \frac{t_t}{\gamma_{mw}}$
- *Governing strength is:* $P_d = \min(P_{wd}, P_{md})$

3. **Weld Size Selection:**

   - Check minimum and maximum permitted weld sizes per IS 800:2007 (*e.g.*, Table 21).
   - Adjust for practical and code-based requirements.

4. **Required Weld Length:**

   - For single-sided: $l_{req} = \frac{P}{P_d}$
   - For double-sided: $l_{req} = \frac{P}{2P_d}$

5. **Length Checks:**

   - Apply long weld reduction factor if $l_{req} > 150\, t_t$.

$$\text{Adjusted } l_{req} = \frac{l_{req}}{\beta_{lw}}, \quad \beta_{lw} = 1.2 - 0.2\, \frac{l_{req}}{150 t_t}$$

   - Ensure weld length is within code limits:

$$l_{eff,min} = \max(4s, 40\,\text{mm}), \quad l_{eff,max} = 70s$$

6. **Connection Detailing:**

   - Compute total connection length, adding end returns ($2s$, not less than 12 mm, at each end).
   - Check required plate overlap and clearances.

7. **Utilization Ratio and Final Checks:**

   - Calculate utilization ratio:

$$\text{Utilization Ratio} = \frac{P}{\text{Design Capacity}}$$

   - Ensure design is within safety and code limits.

Conclusion

The calculation engine successfully implements all required mathematical formulations for welded lap joint connections according to IS 800:2007. The implementation includes:

- Complete mathematical formulations with proper equations
- Robust calculation algorithms with error handling
- Iterative optimization for bolt selection
- Comprehensive capacity reduction factor calculations
- Spacing and arrangement optimization
- Design adequacy verification through utilization ratios

All calculations are implemented with proper validation, error handling, and compliance with the Indian Standard IS 800:2007, ensuring safe and efficient design of welded lap joint connections.

Chapter 7

Conclusions

7.1 Tasks Accomplished

During this internship, I successfully completed three major tasks that significantly enhanced the OSDAG software's functionality and user experience. Each task involved comprehensive analysis, design, implementation, and documentation phases.

7.1.1 GUI Development for Butt Joint Welded

The first task focused on developing a comprehensive graphical user interface for the plate girder module. Key accomplishments included:

- **User Interface Design:** Created an intuitive and user-friendly interface for butt joint welded input and output.
- **Input Field Management:** Implemented various input field types including text boxes, combo boxes, and specialized controls for plate girder parameters.
- **Output Display:** Developed comprehensive output displays showing design results, calculations, and safety checks.
- **User Experience Enhancement:** Designed the interface to provide clear navigation and logical flow for butt joint welded.
- **Integration with Existing System:** Successfully integrated the new GUI components with the existing OSDAG framework.

This task resulted in a functional and user-friendly interface that allows engineers to efficiently design butt joint welded through the OSDAG software platform.

7.1.2 Calculation Implementation for Butt Joint Welded Connection

The second task involved implementing comprehensive calculation logic for butt joint welded connections according to structural engineering standards:

- **Design Standards Implementation:** Implemented calculations following IS 800:2007 standards for butt joint welded connections
- **Weld Capacity Calculations:** Developed algorithms for weld capacity, and combined loading effects
- **Calculations:** Implemented calculations, net area calculations
- **Safety Factor Integration:** Incorporated appropriate safety factors and design checks as per code requirements
- **Validation and Error Handling:** Built comprehensive validation systems to ensure design safety and accuracy

This enhancement provided engineers with reliable calculation tools for designing safe and efficient butt joint welded connections.

7.1.3 Lap Joint Welded

The third task involved extending the existing lap joint welded module to support flat plate sections as a new member type:

- **UI Component Development:** Created dynamic user interface components with conditional field visibility based on section selection.
- **Input Field Management:** Implemented specialized input fields for parameters.
- **Calculation Engine:** Developed a complete calculation system following IS 800:2007 standards for lap joint welded module.

- **Validation Framework:** Built comprehensive validation systems specific to lap joint welded requirements.

This implementation expanded the software's capabilities to handle lap joint welded members, making it more versatile for various engineering applications.

7.2 Skills Developed

Throughout this internship, I developed a comprehensive set of technical and professional skills that will be invaluable for my future career in software development and structural engineering.

7.2.1 Technical Skills

Programming and Software Development

- **Python Programming:** Advanced proficiency in Python development, including object-oriented programming, modular design, and code optimization
- **GUI Development:** Gained expertise in developing graphical user interfaces using PyQt5 framework
- **Software Architecture:** Developed understanding of large-scale software architecture, including module design and component integration
- **Version Control:** Enhanced skills in Git version control and collaborative development practices

Structural Engineering Software

- **OSDAG Platform:** Comprehensive understanding of the OSDAG software architecture and its various modules
- **Design Standards Implementation:** Gained expertise in implementing Indian structural design standards (IS 800:2007) in software
- **Engineering Calculations:** Developed skills in translating complex engineering calculations into efficient software algorithms

- **Connection Design:** Gained understanding of welded connection design principles and calculations

Software Engineering Practices

- **Code Quality:** Developed strong practices in writing clean, maintainable, and well-documented code
- **Testing and Debugging:** Enhanced skills in systematic testing, debugging, and quality assurance
- **User Interface Design:** Learned principles of effective UI/UX design for engineering software
- **Error Handling:** Gained expertise in implementing comprehensive error handling and user feedback systems

Professional Skills

Project Management

- **Task Planning:** Developed ability to break down complex projects into manageable tasks and milestones
- **Time Management:** Enhanced skills in managing multiple tasks simultaneously while meeting deadlines
- **Problem Solving:** Improved analytical and problem-solving skills through tackling complex software engineering challenges
- **Documentation:** Gained expertise in creating comprehensive technical documentation and user guides

Communication and Collaboration

- **Technical Communication:** Improved ability to communicate complex technical concepts clearly and effectively
- **Team Collaboration:** Enhanced skills in working within development teams and coordinating with multiple stakeholders

- **Code Review:** Developed skills in reviewing and providing feedback on others' code
- **Knowledge Sharing:** Learned to effectively share knowledge and best practices with team members

Industry Knowledge

- **Structural Engineering Domain:** Gained deep understanding of structural engineering principles and design processes
- **Software Industry Practices:** Learned industry-standard practices for developing engineering software
- **Quality Assurance:** Developed understanding of quality assurance processes in software development
- **User-Centric Design:** Learned to design software with end-user needs and experience in mind

Personal Development

Adaptability and Learning

- **Rapid Learning:** Enhanced ability to quickly learn new technologies, frameworks, and domain knowledge
- **Adaptability:** Developed flexibility in adapting to changing requirements and project priorities
- **Continuous Improvement:** Cultivated a mindset of continuous learning and skill development

Professional Growth

- **Confidence Building:** Gained confidence in tackling complex technical challenges independently

- **Leadership Skills:** Developed leadership qualities through taking ownership of project components
- **Professional Ethics:** Enhanced understanding of professional responsibility in software development
- **Career Direction:** Gained clarity on career goals and the path forward in software engineering

Impact and Contributions

The work completed during this internship has made significant contributions to the OSDAG software platform:

- **Enhanced User Experience:** The plate girder GUI provides engineers with an intuitive interface for complex plate girder design
- **Expanded Functionality:** Added butt joint bolted connection calculations and flat plate support, making the software more versatile
- **Improved Reliability:** Enhanced calculation accuracy and validation systems ensure safer engineering designs
- **Better Accessibility:** User-friendly interfaces make the software more accessible to engineers of varying experience levels
- **Code Quality:** Contributed to the overall code quality and maintainability of the softwarebase

6.4 Future Recommendations

Based on the experience gained during this internship, several recommendations can be made for future development:

- **GUI Enhancement:** The plate girder interface could be further enhanced with additional visualization features and design aids.
- **Calculation Expansion:** The butt joint bolted connection module could be extended to support additional connection types.

- **Flat Plate Enhancement:** The flat plate implementation could be extended to support additional plate configurations and connection types.
- **Performance Optimization:** Further optimization of calculation algorithms could improve software performance for large-scale projects.
- **Integration Opportunities:** The software could be integrated with other engineering tools and platforms for enhanced workflow.

6.5 Conclusion

This internship has been an invaluable learning experience that has significantly enhanced my technical skills, professional development, and understanding of the intersection between software engineering and structural engineering. The three major tasks completed have not only contributed to the OSDAG software platform but have also provided me with practical experience in real-world software development.

The skills and knowledge gained during this internship will serve as a strong foundation for my future career in software development, particularly in the field of engineering software. The experience of working on a complex, real-world application has provided insights that cannot be gained through academic study alone.

I am grateful for the opportunity to contribute to such an important engineering software platform and look forward to applying the skills and knowledge gained in future professional endeavors. The experience has reinforced my passion for software development and has provided clear direction for my career path forward.

Bibliography

[1] Siddhartha Ghosh, Danish Ansari, Ajmal Babu Mahasrankintakam, Dharma Teja Nuli, Reshma Konjari, M. Swathi, and Subhrajit Dutta. Osdag: A Software for Structural Steel Design Using IS 800:2007. In Sondipon Adhikari, Anjan Dutta, and Satyabrata Choudhury, editors, *Advances in Structural Technologies*, volume 81 of *Lecture Notes in Civil Engineering*, pages 219–231, Singapore, 2021. Springer Singapore.

[2] FOSSEE Project. FOSSEE News - January 2018, vol 1 issue 3. Accessed: 2024-12-05.

[3] FOSSEE Project. Osdag website. Accessed: 2024-12-05.