



FOSSEE Winter Internship Report

On

Code-Based Analysis of a Simply Supported Steel Girder as per IRC Codes using OpenSeesPy

Submitted by

Azhar Khan

Final Year M.Tech (Structure) Student, Department of Civil Engineering

Z H C E T, Aligarh Muslim University

Aligarh U P

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Mr. Ajmal Babu M S

Dr. Nidhi Khare

Mr. Parth Karia

Mr. Ajinkya Dahale

March 30, 2026

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and guided me throughout this internship project. Their encouragement, mentorship, and technical insights played a vital role in the successful completion of my work. I am deeply thankful to the Project Staff of the Osdag Team Ajmal Babu M. S., Dr. Nidhi Khare, Ajinkya Dahale, and Parth Karia for their continuous technical guidance, constructive feedback, and valuable discussions during the course of this project. I extend my heartfelt gratitude to Prof. Siddhartha Ghosh, Principal Investigator of Osdag and Professor in the Department of Civil Engineering at IIT Bombay, for providing me with the opportunity to work on this prestigious project and for his academic support and encouragement. I also sincerely thank Prof. Kannan M. Moudgalya, Principal Investigator of the FOSSEE Project, Department of Chemical Engineering, IIT Bombay, for his vision and leadership in promoting open-source tools for engineering education and research. My appreciation extends to the FOSSEE Managers, Usha Viswanathan and Vineeta Parmar, along with their entire team, for their administrative support and coordination throughout the internship. I gratefully acknowledge the support provided by the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India, for facilitating and funding initiatives that make such academic projects possible. I would also like to thank my fellow interns and colleagues who worked alongside me, for their collaboration, discussions, and a positive learning environment. Finally, I express my sincere gratitude to my college, department faculty members, Head of Department, and Principal for their continuous encouragement and academic support during my studies and internship period.

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	Internship Task 1	11
3.1	Task 1: Problem Statement	11
3.2	Task 1: Tasks Done	11
3.3	Task 1: Python Code	13
3.3.1	Description of the Script	14
3.3.2	Python Code	14
4	Internship Task 2	18
4.1	Task 2: Problem Statement	18
4.2	Task 2: Tasks Done	18
4.3	Full Code	20
5	Conclusions	28
5.1	Tasks Accomplished	28
5.2	Skills Developed	28
5.3	Future Work	29
A	Appendix	30

A.1 Work Reports	30
Bibliography	34

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	SWAYAMPBABHA	Access 24x7 TV programs	Enable SWAYAMPBABHA 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.

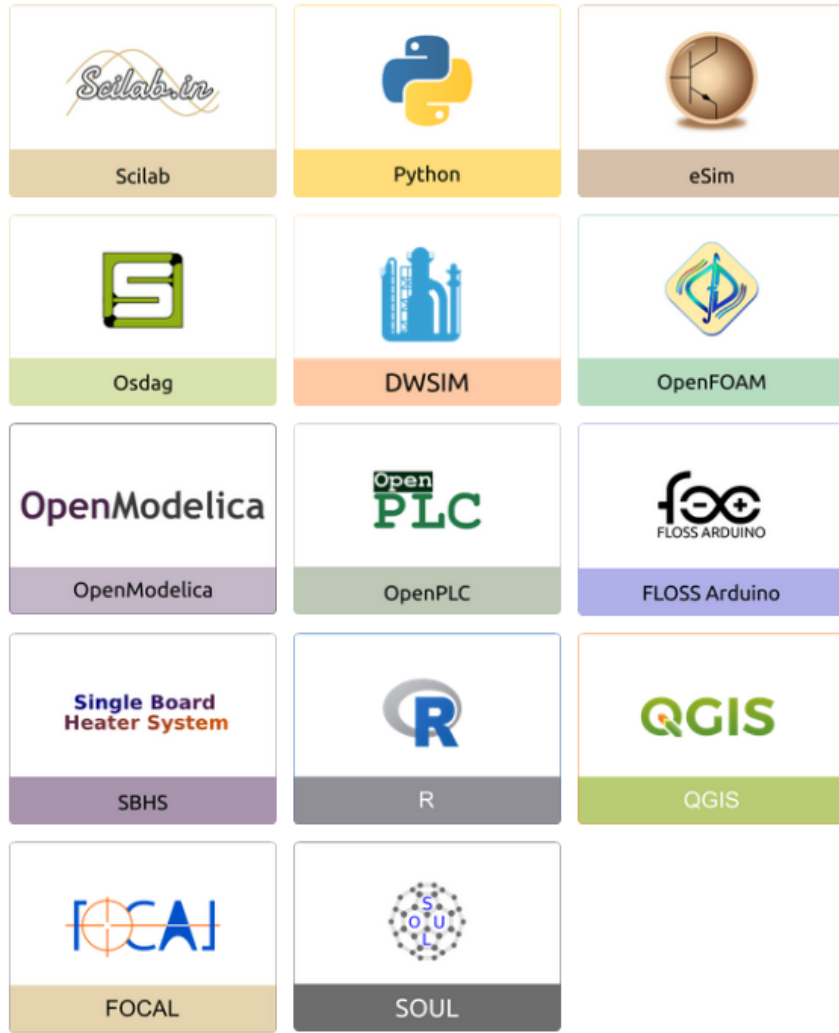


Figure 1.1: FOSSEE Projects and Activities

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 [2, 4, 5]. 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.

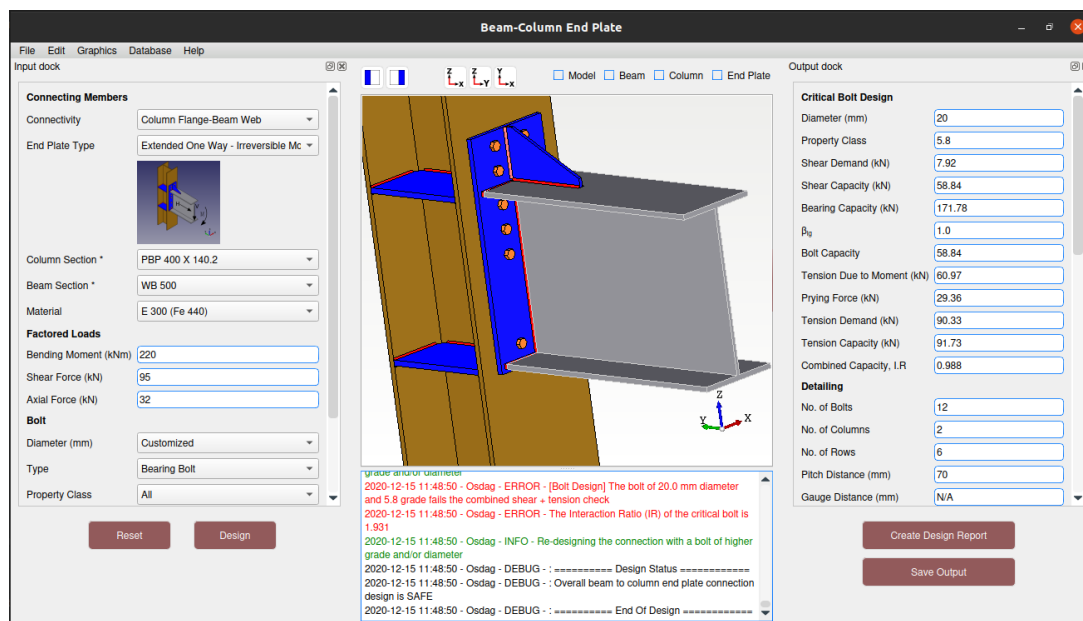


Figure 1.2: Osdag GUI

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

Influence Line Diagram (ILD) for Truss Member Using OpenSeesPy

2.2 Tasks Done

Drawn ILD for member GB

Influence Line Diagram (ILD) for Truss Member Using OpenSeesPy

You are required to write a **Python**
program that:

- Uses **OpenSeesPy** for structural analysis.
- Computes the **influence line** for the **axial force in a selected truss member**.
- Plots the **Influence Line Diagram (ILD)** as a result of a **unit moving load**.

Figure 2.1: problem for screening task

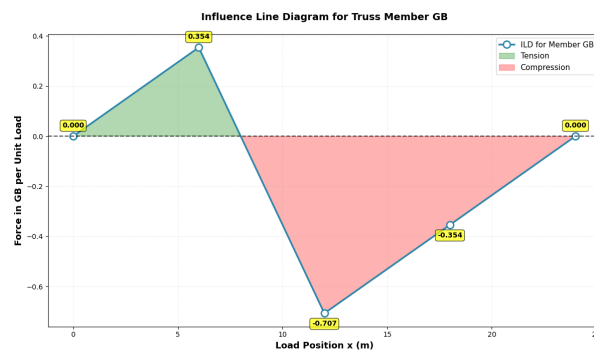


Figure 2.2: ILD for member GB

Chapter 3

Internship Task 1

3.1 Task 1: Problem Statement

Make Python script to get BMD and SFD results for Steel Plate girder

3.2 Task 1: Tasks Done

- Idealized the 35 m steel girder bridge as a 2D simply supported beam model for linear static analysis.
- Discretized the span into 50 beam elements to ensure accurate force and displacement distribution.
- Generated nodal coordinates along the span at uniform spacing ($x = L / nEle$).
- Defined structural model in OpenSeesPy using a 2D frame formulation with 3 DOF per node (U_x , U_y , Rotation).
- Assigned linear elastic material properties ($E = 200 \text{ GPa}$) representing structural steel behavior.
- Defined sectional properties (A and I) corresponding to the equivalent composite girder stiffness.
- Selected elasticBeamColumn element to model combined axial and flexural behavior.

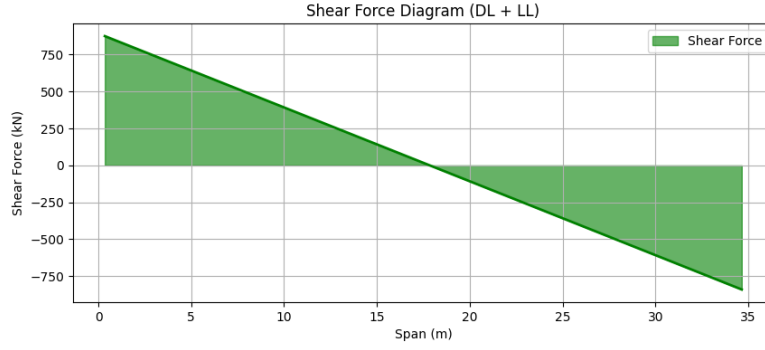


Figure 3.1: Shear Force due to DL LL

- Applied geometric transformation (Linear) to relate local and global coordinate systems.
- Modeled boundary conditions:
 - Left support: restrained in X and Y (pinned)
 - Right support: restrained only in Y (roller)
- Applied Dead Load (DL) as uniformly distributed load (UDL) representing self-weight and deck weight as per IRC:6-2017 [3].
- Applied Live Load (LL) as equivalent UDL based on simplified IRC loading representation.
- Defined separate load patterns for DL and LL using linear time series for static load application.
- Performed linear static analysis using load control integrator and direct stiffness solution method.
- Extracted element end forces using `ops.eleForce()` to compute: Shear Force values Bending Moment values along the span.
- Extracted nodal displacements and support reactions using `ops.nodeDisp()` and `ops.nodeReaction()` to validate equilibrium and generate SFD, BMD, and deflection diagrams.

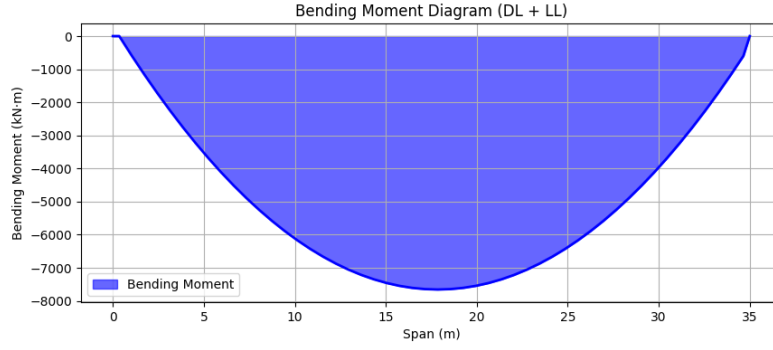


Figure 3.2: Bending Moment diagram due to DL LL

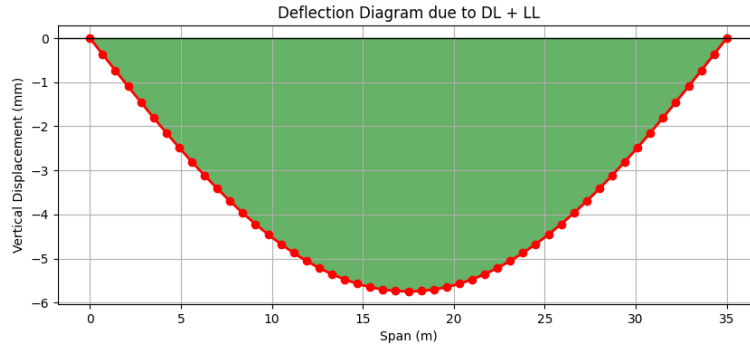


Figure 3.3: Deflection diagram due to DL and LL

Description	Value
Left support vertical reaction	875.0 kN
Right support vertical reaction	875.0 kN
Total reaction	1750.0 kN
Maximum DL + LL deflection	-5.7469 mm

Table 3.1: DL + LL Reactions and Deflection

3.3 Task 1: Python Code

This section presents the Python-based finite element script developed during the internship for the analysis of a simply supported steel girder RC deck bridge using OpenSeesPy. The script models the bridge geometry, assigns material and sectional properties, defines boundary conditions, and applies loading as per IRC:6-2017. It automates the computation of support reactions, shear force diagram (SFD), bending moment diagram (BMD), vertical deflection under gravity loads. The developed code ensures systematic implementation of code-based loading and provides a reliable computational framework for bridge response evaluation.

3.3.1 Description of the Script

The script is structured as follows:

- **Input Parameters** The user defines key structural parameters including span length, number of elements for discretization, material properties (Young's modulus), sectional properties (area and moment of inertia), boundary conditions, and uniformly distributed Dead Load (DL) and Live Load (LL) values as per IRC:6-2017.
- **Modeling and Analysis** The program generates the finite element model in OpenSeesPy, assigns material and section properties, applies support conditions, and imposes DL and LL as uniform beam loads. Linear static analysis is then performed using the stiffness method to compute nodal displacements and element forces.
- **Output** The script automatically extracts support reactions, shear forces, bending moments, and vertical deflection values, and generates Shear Force Diagram (SFD), Bending Moment Diagram (BMD), and deflection plots for structural response evaluation.

3.3.2 Python Code

The Python script is shown below. Each section is commented for clarity.

Listing 3.1: Extract BMD, SFD and Deflection Diagram

```
1 %-----begin code-----
2
3 import openseespy.opensees as ops
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 # -----
8 # 1. MODEL DEFINITION
9 # -----
10 ops.wipe()
11 ops.model('basic', '-ndm', 2, '-ndf', 3)
12
```

```

13 # -----
14 # 2. GEOMETRY
15 # -----
16 L = 35 # length of span
17 nEle = 50
18 dx = L / nEle
19
20 for i in range(nEle + 1):
21     ops.node(i + 1, i * dx, 0.0)
22
23 # Supports (Simply Supported      longitudinally fixed at left)
24 ops.fix(1, 1, 1, 0)                # [Left (Pin) support]
25 ops.fix(nEle + 1, 0, 1, 0)         # Expansion bearing [Right(roller)
    support]
26
27 # -----
28 # 3. MATERIAL & SECTION
29 # -----
30 E = 200e9
31 A = 0.02
32 I = 8.5e-4
33
34 ops.geomTransf('Linear', 1)
35
36 for i in range(nEle):
37     ops.element('elasticBeamColumn',
38                 i + 1, i + 1, i + 2, A, E, I, 1)
39
40 # -----
41 # 4. LOADS
42 # -----
43
44 # ---- DEAD LOAD (IRC 6) ----
45 w_DL = -30.0 # kN/m (assumed value)
46
47 ops.timeSeries('Linear', 1)
48 ops.pattern('Plain', 1, 1)
49
50 for i in range(nEle):

```

```

51     ops.eleLoad('-ele', i + 1, '-type', '-beamUniform', w_DL)
52
53 # ---- LIVE LOAD (IRC Class A      simplified UDL) ----
54 w_LL = -20.0 # kN/m (assumed value applied as udl)
55
56 ops.timeSeries('Linear', 2)
57 ops.pattern('Plain', 2, 2)
58
59 for i in range(nEle):
60     ops.eleLoad('-ele', i + 1, '-type', '-beamUniform', w_LL)
61
62 # -----
63 # 5. ANALYSIS 1: DL + LL      to get  BMD & SFD
64 # -----
65
66 ops.wipeAnalysis()
67
68 ops.system('BandGeneral')
69 ops.numberer('Plain')
70 ops.constraints('Plain')
71 ops.integrator('LoadControl', 1.0)
72 ops.algorithm('Linear')
73 ops.analysis('Static')
74 ops.analyze(1)
75
76
77 # -----
78 # 6. DISPLACEMENT DUE TO DL + LL
79 # -----
80 x_def = []
81 v_def = []
82
83 for i in range(1, nEle + 2):
84     x_def.append((i - 1) * dx)
85     v_def.append(ops.nodeDisp(i, 2)) # Vertical displacement (Y)
86
87 v_def_mm = [v * 1000 for v in v_def]
88
89

```

```

90 # ---- Extract SFD & BMD ----
91 x = []
92 V = []
93 M = []
94
95 for i in range(nEle):
96     f = ops.eleForce(i + 1)
97     V.append(f[1])
98     M.append(f[2])
99     x.append((i + 0.5) * dx)
100
101 x_plot = [0] + x + [L]
102 M_plot = [0] + M + [0]
103 # -----
104 # 7. SUPPORT REACTIONS DUE TO DL + LL
105 # -----
106 ops.reactions()
107
108 R1_DL_LL = ops.nodeReaction(1, 2)          # Left support, vertical
109 R2_DL_LL = ops.nodeReaction(nEle+1, 2)     # Right support, vertical
110
111 print("DL + LL Reactions:")
112 print("Left support vertical reaction =", round(R1_DL_LL, 2), "kN")
113 print("Right support vertical reaction =", round(R2_DL_LL, 2), "kN")
114 print("Total reaction =", round(R1_DL_LL + R2_DL_LL, 2), "kN")
115
116 print("Maximum DL + LL deflection =",
117       round(min(v_def_mm), 4), "mm")
118
119 %----- end code -----

```

Chapter 4

Internship Task 2

4.1 Task 2: Problem Statement

Apply Seismic load according to IS 1893:2016 to get Deflection and reaction

4.2 Task 2: Tasks Done

- Adopted the Equivalent Static Method for seismic analysis as per IS 1893 (Part 1):2016.
- Considered seismic zone factor (Z), importance factor (I), response reduction factor (R), and spectral acceleration coefficient (S_a/g) as per code provisions.
- Calculated the horizontal seismic coefficient using [1]

$$A_h = \frac{Z}{2} * \frac{I}{R} * \frac{S_a}{g}$$

- Determined the seismic weight (W) of the bridge using total dead load over the span length.
- Computed the design base shear (V_b) using:

$$V_b = A_h * W$$

- Removed previously applied vertical load patterns (DL LL) to isolate seismic response.

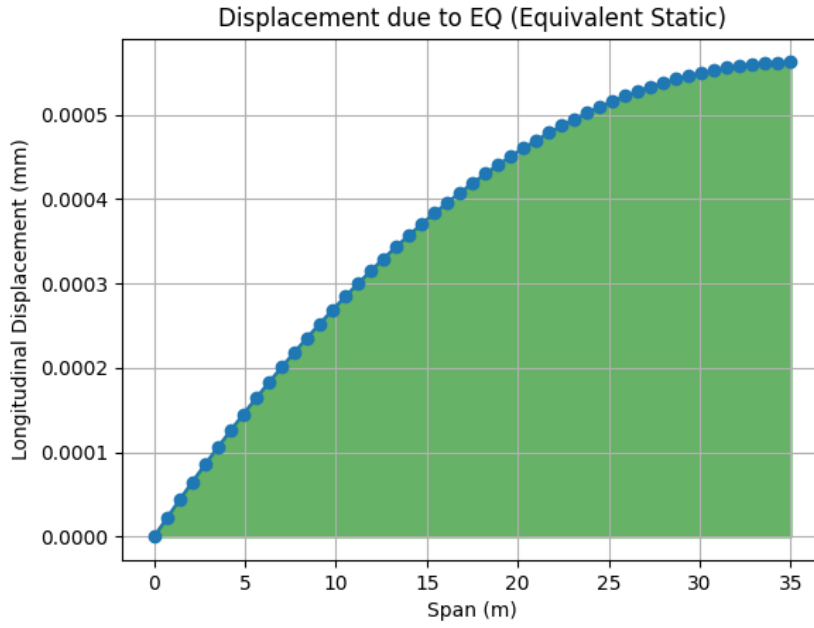


Figure 4.1: Deflection due to EQ load

- Applied the calculated base shear as equivalent horizontal nodal forces distributed uniformly along the span.
- Linear static analysis performed under horizontal loading using the stiffness-based solution approach.
- Extracted longitudinal nodal displacements to determine maximum seismic deflection along the span.
- Obtained horizontal support reactions using `nodeReaction()` and verified that the sum of reactions equals the computed base shear, satisfying global equilibrium.

Following the evaluation of structural response under vertical loading (Dead and Live Loads) as per IRC:6-2017, the analysis was extended to assess the bridge performance under seismic action using the Equivalent Static Method prescribed in IS 1893 (Part 1):2016. While Task 1 focused on flexural behavior, shear distribution, and vertical deflection, Task 2 examined horizontal displacement and support reactions resulting from the computed design base shear. This sequential approach enabled a comprehensive understanding of both gravity and seismic response characteristics of the bridge model.

Description	Value	Unit
Left support horizontal reaction	126.0	kN
Right support horizontal reaction	0.0	kN
Sum of reactions	126.0	kN
Ah	0.12	-
W	1050	-
Vb	126.0	-

Table 4.1: EQ Reactions

4.3 Full Code

listings

```
# -----
# Steel Girder RC Deck Bridge      Single Girder
# Span = 35 m (Assuemed as Simply Supported)
#
# DL & LL : IRC:6 2017
# EQ Load : IS 1893 (Part 1): 2016      Equivalent Static Method
#
# OUTPUT:
# 1. Reactions due to DL
# 1. BMD & SFD due to DL + LL
# 2. Horizontal displacement & reaction due to EQ load
# -----

import openseespy.opensees as ops
import numpy as np
import matplotlib.pyplot as plt

# -----
# 1. MODEL DEFINITION
# -----

ops.wipe()
ops.model('basic', '-ndm', 2, '-ndf', 3)
```

```

# -----
# 2. GEOMETRY
# -----

L = 35 # length of span
nEle = 50
dx = L / nEle

for i in range(nEle + 1):
    ops.node(i + 1, i * dx, 0.0)

# Supports (Simply Supported      longitudinally fixed at left)
ops.fix(1, 1, 1, 0)                # [Left (Pin) support]
ops.fix(nEle + 1, 0, 1, 0)         # Expansion bearing [Right(
    roller) support]

# -----
# 3. MATERIAL & SECTION
# -----

E = 200e9
A = 0.02
I = 8.5e-4

ops.geomTransf('Linear', 1)

for i in range(nEle):
    ops.element('elasticBeamColumn',
                i + 1, i + 1, i + 2, A, E, I, 1)

# -----
# 4. LOADS
# -----

# ---- DEAD LOAD (IRC 6) ----
w_DL = -30.0 # kN/m (assumed value)

```

```

ops.timeSeries('Linear', 1)
ops.pattern('Plain', 1, 1)

for i in range(nEle):
    ops.eleLoad('-ele', i + 1, '-type', '-beamUniform', w_DL)

# ---- LIVE LOAD (IRC Class A      simplified UDL) ----
w_LL = -20.0 # kN/m (assumed value applied as udl)

ops.timeSeries('Linear', 2)
ops.pattern('Plain', 2, 2)

for i in range(nEle):
    ops.eleLoad('-ele', i + 1, '-type', '-beamUniform', w_LL)

# -----
# 5. ANALYSIS 1: DL + LL      to get  BMD & SFD
# -----

ops.wipeAnalysis()

ops.system('BandGeneral')
ops.numberer('Plain')
ops.constraints('Plain')
ops.integrator('LoadControl', 1.0)
ops.algorithm('Linear')
ops.analysis('Static')
ops.analyze(1)

# -----
# 6. DISPLACEMENT DUE TO DL + LL
# -----

```

```

x_def = []
v_def = []

for i in range(1, nEle + 2):
    x_def.append((i - 1) * dx)
    v_def.append(ops.nodeDisp(i, 2))    # Vertical displacement (Y
    )

v_def_mm = [v * 1000 for v in v_def]

# ---- Extract SFD & BMD ----
x = []
V = []
M = []

for i in range(nEle):
    f = ops.eleForce(i + 1)
    V.append(f[1])
    M.append(f[2])
    x.append((i + 0.5) * dx)

x_plot = [0] + x + [L]
M_plot = [0] + M + [0]

# -----
# 7. SUPPORT REACTIONS DUE TO DL + LL
# -----

ops.reactions()

R1_DL_LL = ops.nodeReaction(1, 2)    # Left support,
    vertical
R2_DL_LL = ops.nodeReaction(nEle+1, 2)    # Right support,
    vertical

```

```

print("DL + LL Reactions:")
print("Left support vertical reaction =", round(R1_DL_LL, 2), "kN")
print("Right support vertical reaction =", round(R2_DL_LL, 2), "kN")
print("Total reaction =", round(R1_DL_LL + R2_DL_LL, 2), "kN")

print("Maximum DL + LL deflection =",
      round(min(v_def_mm), 4), "mm")

# -----
# 8. EARTHQUAKE LOAD      IS 1893 (EQUIVALENT STATIC method)
# -----

ops.remove('loadPattern', 1) # DL
ops.remove('loadPattern', 2) # LL
ops.loadConst('-time', 0.0)

Z = 0.24
I_factor = 1.2
R = 3.0
Sa_g = 2.5

Ah = (Z / 2) * (I_factor / R) * Sa_g

W = 30 * L # kN (same DL used as seismic weight)
Vb = Ah * W # Base shear (kN)

# -----
# 9. STATIC EQ LOAD (Equivalent Static Method)
# -----

ops.timeSeries('Linear', 10)
ops.pattern('Plain', 10, 10)

Fx = Vb

```

```

# Apply horizontal load at all nodes proportionally
for i in range(2, nEle+2):
    ops.load(i, Fx/nEle, 0.0, 0.0)

# -----
# 10. ANALYSIS 2: EQ ONLY (STATIC)
# -----

ops.wipeAnalysis()

ops.system('BandGeneral')
ops.numberer('Plain')
ops.constraints('Plain')
ops.integrator('LoadControl', 1.0)
ops.algorithm('Linear')
ops.analysis('Static')

ops.analyze(1)

# ---- Displacement due to EQ ----
x_disp = []
u_disp = []

for i in range(1, nEle + 2):
    x_disp.append((i - 1) * dx)
    u_disp.append(ops.nodeDisp(i, 1))

ops.reactions()

Rx_left = ops.nodeReaction(1, 1)
Rx_right = ops.nodeReaction(nEle+1, 1)

print("\nEQ Reactions:")

```

```

print("Left support horizontal reaction =", round(Rx_left, 2), "
      kN")
print("Right support horizontal reaction =", round(Rx_right, 2),
      "kN")
print("Sum of reactions =", round(Rx_left + Rx_right, 2), "kN")
print("Ah =", round(Ah,5))
print("W =", W)
print("Vb =", round(Vb,3) "kN")

# -----
# 11. PLOTS
# -----

# --- SFD ---
plt.figure(figsize=(10, 4))
plt.fill_between(x, V, 0, color='green', alpha=0.6, label='Shear
      Force')
plt.plot(x, V, color='green', linewidth=2)
plt.xlabel("Span (m)")
plt.ylabel("Shear Force (kN)")
plt.title("Shear Force Diagram (DL + LL)")
plt.legend()
plt.grid()
plt.show()

# --- BMD ---
plt.figure(figsize=(10, 4))
plt.fill_between(x_plot, M_plot, 0, color='blue', alpha=0.6,
      label='Bending Moment')
plt.plot(x_plot, M_plot, color='blue', linewidth=2)
plt.xlabel("Span (m)")
plt.ylabel("Bending Moment (kN m)")
plt.title("Bending Moment Diagram (DL + LL)")
plt.legend()

```

```

plt.grid()
plt.show()

# --- EQ Displacement ---
plt.figure()
u_disp_mm = [u * 1000 for u in u_disp]
plt.fill_between(x_disp, u_disp_mm, 0, color='green', alpha=0.6,
                 label='Displacement')
plt.plot(x_disp, u_disp_mm, marker='o')
plt.xlabel("Span (m)")
plt.ylabel("Longitudinal Displacement (mm)")
plt.title("Displacement due to EQ (Equivalent Static)")
plt.grid()
plt.show()

# Displacement due to DL+LL
plt.figure(figsize=(10, 4))
plt.fill_between(x_def, v_def_mm, 0, color='green', alpha=0.6,
                 label='Displacement')
plt.plot(x_def, v_def_mm, marker='o', color='red', linewidth=2)
plt.axhline(0, color='black', linewidth=1)
plt.xlabel("Span (m)")
plt.ylabel("Vertical Displacement (mm)")
plt.title("Deflection Diagram due to DL + LL")
plt.grid(True)
plt.show()

print("Maximum EQ displacement =",
      round(max(abs(np.array(u_disp_mm))), 6), "mm")

```

Chapter 5

Conclusions

5.1 Tasks Accomplished

The internship work focused on the numerical modeling and structural analysis of a 35 m simply supported steel girder RC deck bridge using OpenSeesPy. Vertical load analysis was carried out in accordance with IRC:6-2017 to obtain bending moment, shear force, support reactions, and deflection profiles. The results showed agreement proper with classical beam theory, validating the finite element modeling approach. Further, seismic analysis was performed using the Equivalent Static Method as per IS 1893 (Part 1):2016 to compute design base shear, horizontal displacements, and reactions. Overall, the study demonstrated effective application of code-based loading and computational modeling for bridge response evaluation.

5.2 Skills Developed

1. Structural steel analysis and modeling.
2. Bridge structural behavior assessment under IRC loading.
3. Finite element modeling using OpenSeesPy.
4. Python Programming for Bridge engineering using openseesPy Library for structural simulation and result extraction.
5. Seismic load calculation and application using Equivalent Static Method.

5.3 Future Work

- The model can be extended to a multi-girder bridge system for better load distribution analysis.
- Transverse members such as cross girders can be added to improve lateral stability.
- The analysis can be upgraded to a 3D model to capture torsional and realistic behavior.
- Moving load analysis can be included instead of simplified UDL loading.
- Time-history analysis can be performed for more accurate seismic response.
- Nonlinear analysis can be carried out to study actual structural behavior under higher loads.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report			
Name:	Azhar Khan		
Project:	Osdagbridge		
Internship	FOSEE Winter Fellowship		
DATE	DAY	TASK	Hours Worked
01-Dec-2025	Monday	Studied project objectives and reviewed bridge modeling concepts	6
02-Dec-2025	Tuesday	Literature review on IRC loading provisions	6
03-Dec-2025	Wednesday	Installed and configured OpenSeesPy environment	5
04-Dec-2025	Thursday	Planned modeling strategy for simply supported girder	6
05-Dec-2025	Friday	Developed initial geometry generation script	6
06-Dec-2025	Saturday	Implemented nodal discretization and element connectivity	5
07-Dec-2025	Sunday	—	—
08-Dec-2025	Monday	Assigned material and sectional properties	6
09-Dec-2025	Tuesday	Applied boundary conditions (pinned & roller supports)	6
10-Dec-2025	Wednesday	Verified structural model stability	5
11-Dec-2025	Thursday	Applied Dead Load as UDL	6
12-Dec-2025	Friday	Applied Live Load as per IRC provisions	6
13-Dec-2025	Saturday	Debugged load application issues	5
14-Dec-2025	Sunday	—	—
15-Dec-2025	Monday	Performed static analysis for DL	6
16-Dec-2025	Tuesday	Performed combined DL + LL analysis	6
17-Dec-2025	Wednesday	Extracted support reactions	5
18-Dec-2025	Thursday	Extracted shear force values from elements	6
19-Dec-2025	Friday	Extracted bending moment values	6

20-Dec-2025	Saturday	Generated Shear Force Diagram (SFD)	5
21-Dec-2025	Sunday	—	—
22-Dec-2025	Monday	Generated Bending Moment Diagram (BMD)	6
23-Dec-2025	Tuesday	Computed vertical deflection profile	6
24-Dec-2025	Wednesday	Compared FEM results with theoretical beam equations	5
25-Dec-2025	Thursday	Improved plotting and result visualization	5
26-Dec-2025	Friday	Organized code structure and added comments	6
27-Dec-2025	Saturday	Validated equilibrium and reaction sums	5
28-Dec-2025	Sunday	—	—
29-Dec-2025	Monday	Documented methodology for vertical load analysis	6
30-Dec-2025	Tuesday	Prepared tables and diagrams for report	6
31-Dec-2025	Wednesday	Reviewed and refined DL + LL analysis section	5
01-Jan-2026	Thursday	Studied seismic provisions (IS 1893 Equivalent Static Method)	5
02-Jan-2026	Friday	Calculated seismic coefficient (A_h)	6
03-Jan-2026	Saturday	Computed seismic weight and base shear	6
04-Jan-2026	Sunday	—	—
05-Jan-2026	Monday	Removed gravity load patterns and prepared seismic model	6
06-Jan-2026	Tuesday	Applied equivalent horizontal loads	6
07-Jan-2026	Wednesday	Performed static seismic analysis	6
08-Jan-2026	Thursday	Extracted horizontal displacements	5
09-Jan-2026	Friday	Extracted horizontal support reactions	5
10-Jan-2026	Saturday	Verified base shear equilibrium	5
11-Jan-2026	Sunday	—	—
12-Jan-2026	Monday	Generated displacement plots	6
13-Jan-2026	Tuesday	Compared gravity vs seismic response	6
14-Jan-2026	Wednesday	Refined seismic load implementation	5
15-Jan-2026	Thursday	Improved code modularization	6
16-Jan-2026	Friday	Optimized plotting routines	5
17-Jan-2026	Saturday	Documented seismic methodology	6
18-Jan-2026	Sunday	—	—
19-Jan-2026	Monday	Prepared technical descriptions and flowcharts	6
20-Jan-2026	Tuesday	Added validation calculations	6

21-Jan-2026	Wednesday	Organized internship report sections	5
22-Jan-2026	Thursday	Cross-checked numerical results	6
23-Jan-2026	Friday	Reviewed modeling assumptions	5
24-Jan-2026	Saturday	Incorporated corrections after review	5
25-Jan-2026	Sunday		
26-Jan-2026	Monday		
27-Jan-2026	Tuesday	Updated figures and formatting	5
28-Jan-2026	Wednesday	Final review of analytical results	5
29-Jan-2026	Thursday	Compiled complete documentation	5
30-Jan-2026	Friday	Internal verification of all calculations	4
31-Jan-2026	Saturday	Final content proofreading and formatting	4

Bibliography

- [1] Bureau of Indian Standards. IS 1893 (Part 1): 2016 - Criteria for Earthquake Resistant Design of Structures, Part 1: General Provisions and Buildings. Standard, Bureau of Indian Standards, New Delhi, India, 2016.
- [2] 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.
- [3] Indian Roads Congress. *Standard Specifications and Code of Practice for Road Bridges, Section II: Loads and Stresses*. Indian Roads Congress, New Delhi, India, 2017. IRC:6-2017, Fourth Revision.
- [4] FOSSEE Project. FOSSEE News - January 2018, vol 1 issue 3. Accessed: 2024-12-05.
- [5] FOSSEE Project. Osdag website. Accessed: 2024-12-05.