Visualization of Structural Force Diagrams Using Xarray and Plotly

OSDAG Screening Task – Software Development

Submitted for:

OSDAG (Open Steel Design and Graphics)

FOSSEE, IIT Bombay

Submitted by:

Divit Singhania

B.Tech Computer Science (AI & ML)

1. Introduction

Structural analysis software such as MIDAS provides advanced post-processing tools to visualize internal forces like shear force and bending moment in both two-dimensional and three-dimensional formats.

The objective of this screening task is to replicate similar post-processing capabilities using Python, based on internal force results stored in an Xarray dataset. The task involves:

- Extracting internal force data from an Xarray dataset

- Generating 2D Shear Force Diagrams (SFD) and Bending Moment Diagrams (BMD)

- Creating 3D MIDAS-style visualizations for all longitudinal girders using actual node coordinates and element connectivity

2. Dataset and Supporting Files

2.1 Xarray Dataset (screening_task.nc)

The provided Xarray dataset contains internal force results of a grillage bridge model.

Dimensions:

- Element: Beam element IDs

- Component: Internal force components

Data Variable:

- forces (Element, Component)

Relevant components used:

- Vy_i, Vy_j → Shear force at start and end nodes

- Mz_i, Mz_j → Bending moment at start and end nodes

The suffix:

- _i denotes the start node of an element

- _j denotes the end node of an element

```
===== Step 1: Dataset Loaded Successfully =====
Identified component: Vy_i
Identified component: Vy_j
Identified component: Mz_i
Identified component: Mz_j

===== Step 2: Extracted Central Girder Data =====
<xarray.DataArray 'forces' (Element: 9, Component: 4)> Size: 288B
[36 values with dtype=float64]
Coordinates:
  * Element    (Element) int32 36B 15 24 33 42 51 60 69 78 83
  * Component  (Component) object 32B 'Vy_i' 'Vy_j' 'Mz_i' 'Mz_j'
    Loadcase   object 8B ...

Tabular View of Extracted Data:
          Loadcase                ... Force_Value
Component     Vy_i   Vy_j   Mz_i  ...        Vy_j          Mz_i          Mz_j
Element                           ...
15           point  point  point  ...    2.277489  1.190159e-13 -6.326410e+00
24           point  point  point  ...    2.439648  6.326410e+00 -1.310326e+01
33           point  point  point  ...    2.639270  1.310326e+01 -2.043436e+01
42           point  point  point  ...    0.919836  2.043436e+01 -2.298948e+01
51           point  point  point  ...   -1.985087  2.298948e+01 -1.747531e+01
60           point  point  point  ...   -1.558680  1.747531e+01 -1.314561e+01
69           point  point  point  ...   -1.534609  1.314561e+01 -8.882924e+00
78           point  point  point  ...   -1.646929  8.882924e+00 -4.308085e+00
83           point  point  point  ...   -1.550898  4.308085e+00 -1.314504e-13

[9 rows x 8 columns]

Process finished with exit code 0
```

Figure 1: Xarray dataset structure showing dimensions and force components

2.2 Node Coordinates and Element Connectivity

node.py:

node_id → (x, y, z)

element.py:

element_id → (start_node, end_node)

node.py

```
# Node coordinates: node_id : [x, y, z]
nodes = {
    1: [0.0000, 0.0000, 0.0000],
    2: [0.0000, 0.0000, 1.2000],
    3: [0.0000, 0.0000, 5.1750],
    4: [0.0000, 0.0000, 9.1500],
    5: [0.0000, 0.0000, 10.3500],
    6: [25.0000, 0.0000, 0.0000],
    7: [25.0000, 0.0000, 1.2000],
    8: [25.0000, 0.0000, 5.1750],
    9: [25.0000, 0.0000, 9.1500],
    10: [25.0000, 0.0000, 10.3500],
    11: [2.7778, 0.0000, 0.0000],
    12: [2.7778, 0.0000, 1.2000],
    13: [2.7778, 0.0000, 5.1750],
    14: [2.7778, 0.0000, 9.1500],
    15: [2.7778, 0.0000, 10.3500],
    16: [5.5556, 0.0000, 0.0000],
    17: [5.5556, 0.0000, 1.2000],
    18: [5.5556, 0.0000, 5.1750],
    19: [5.5556, 0.0000, 9.1500],
    20: [5.5556, 0.0000, 10.3500],
    21: [8.3333, 0.0000, 0.0000],
    22: [8.3333, 0.0000, 1.2000],
    23: [8.3333, 0.0000, 5.1750],
    24: [8.3333, 0.0000, 9.1500],
    25: [8.3333, 0.0000, 10.3500],
    26: [11.1111, 0.0000, 0.0000],
    27: [11.1111, 0.0000, 1.2000],
    28: [11.1111, 0.0000, 5.1750],
    29: [11.1111, 0.0000, 9.1500],
    30: [11.1111, 0.0000, 10.3500],
    31: [13.8889, 0.0000, 0.0000],
    32: [13.8889, 0.0000, 1.2000],
```

element.py

```
# member_id : [start_node_id, end_node_id]
members = {
    15: [3, 13],
    24: [13, 18],
    33: [18, 23],
    42: [23, 28],
    51: [28, 33],
    60: [33, 38],
    69: [38, 43],
    78: [43, 48],
    83: [48, 8],
    14: [2, 12],
    23: [12, 17],
    32: [17, 22],
    41: [22, 27],
    50: [27, 32],
    59: [32, 37],
    68: [37, 42],
    77: [42, 47],
    82: [47, 7],
    16: [4, 14],
    25: [14, 19],
    34: [19, 24],
    43: [24, 29],
    52: [29, 34],
    61: [34, 39],
    70: [39, 44],
    79: [44, 49],
    84: [49, 9],
    13: [1, 11],
    22: [11, 16],
    31: [16, 21],
    40: [21, 26],
    49: [26, 31],
```

Figure 2: Example of node coordinates and element connectivity

3. Task-1: 2D Shear Force and Bending Moment Diagrams

The 2D diagrams show continuous shear force and bending moment variation along the central longitudinal girder using the sign convention stored in the dataset.

Central Longitudinal Girder:

[15, 24, 33, 42, 51, 60, 69, 78, 83]

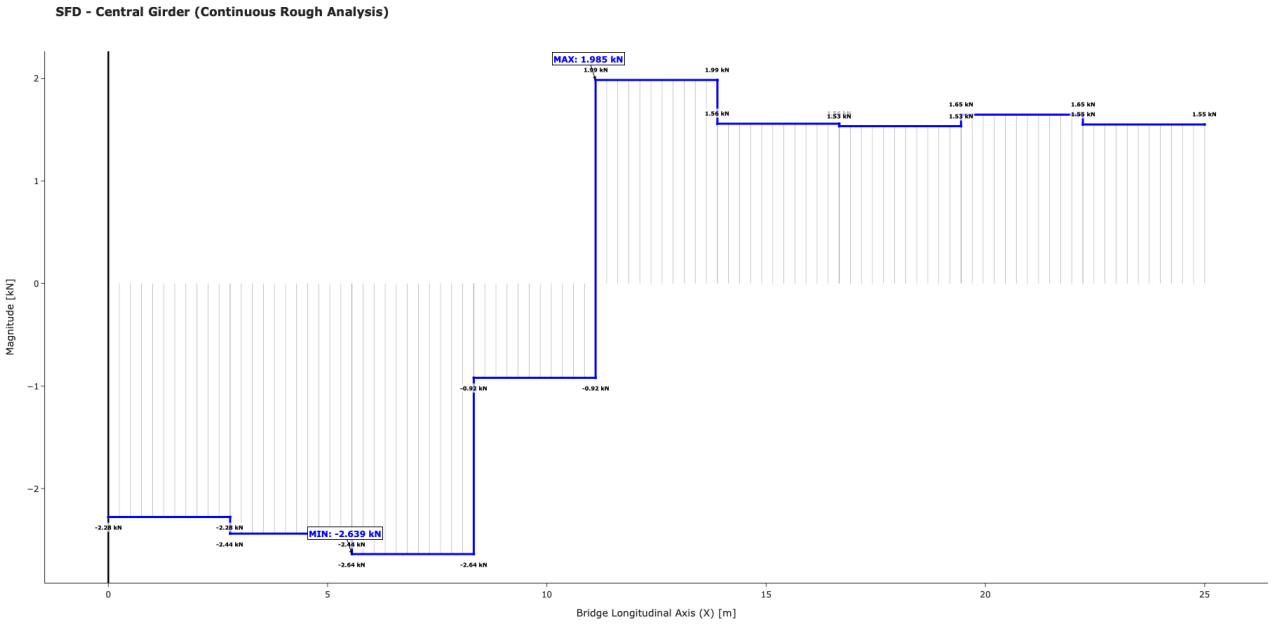

Figure 3: 2D Shear Force Diagram (SFD)

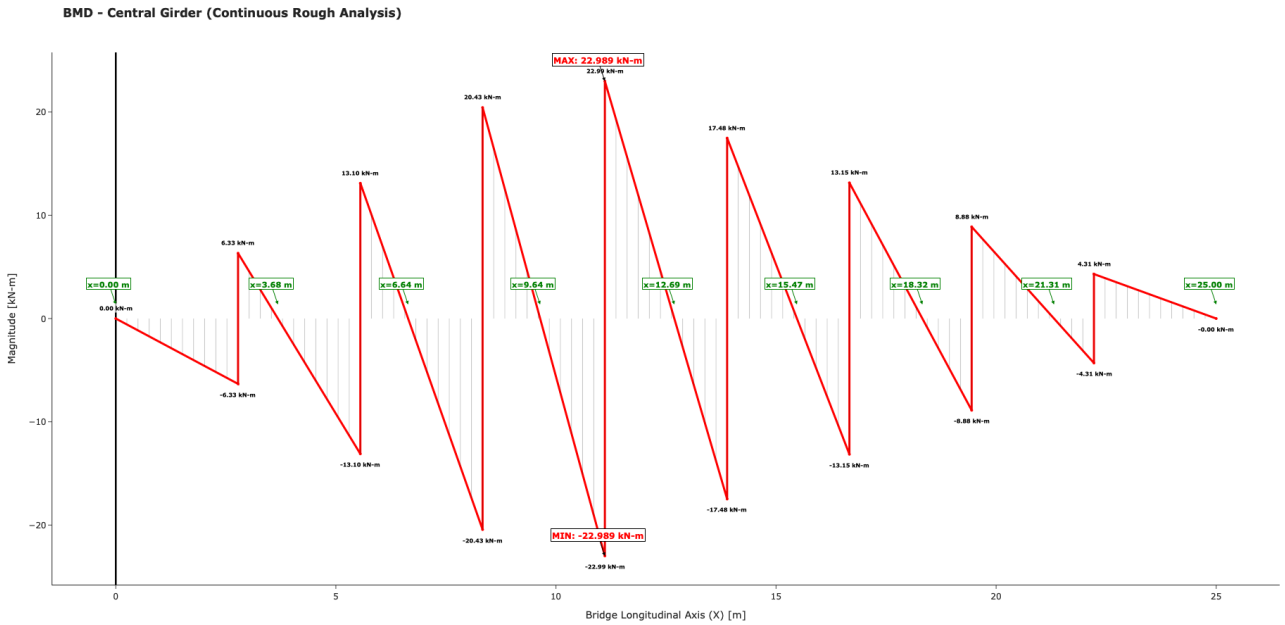

Figure 4: 2D Bending Moment Diagram (BMD)

4. Task-2: 3D Shear Force and Bending Moment Diagrams

The 3D diagrams represent MIDAS-style post-processing with force magnitudes extruded vertically along the bridge framing.

Five girders visualized using exact node and element definitions.

Girder 1 – Elements ->[13, 22, 31, 40, 49, 58, 67, 76, 81], Node -> [1, 11, 16, 21, 26, 31, 36, 41, 46, 6]

Girder 2 – Elements ->[14, 23, 32, 41, 50, 59, 68, 77, 82], Node -> [2, 12, 17, 22, 27, 32, 37, 42, 47, 7]

Girder 3 – Elements ->[15, 24, 33, 42, 51, 60, 69, 78, 83], Node -> [3, 13, 18, 23, 28, 33, 38, 43, 48, 8]

Girder 4 – Elements ->[16, 25, 34, 43, 52, 61, 70, 79, 84], Node -> [4, 14, 19, 24, 29, 34, 39, 44, 49, 9]

Girder 5 – Elements -> [17, 26, 35, 44, 53, 62, 71, 80, 85], Node -> [5, 15, 20, 25, 30, 35, 40, 45, 50, 10]

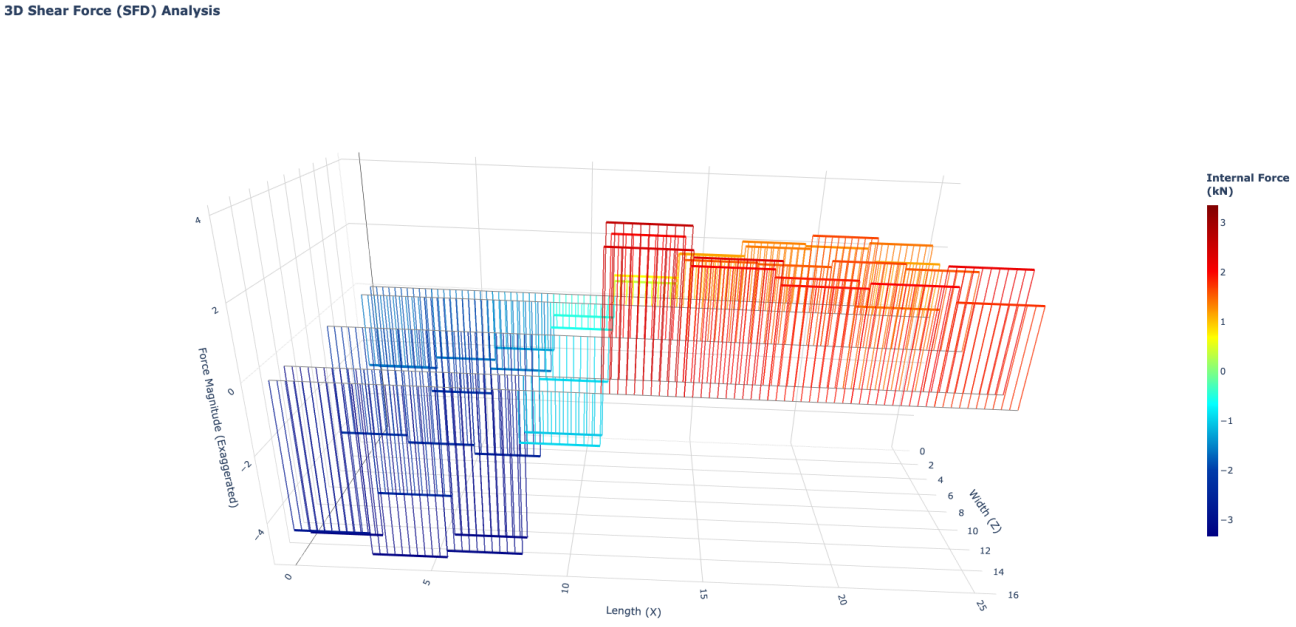

Figure 5: 3D Shear Force Diagram (SFD)

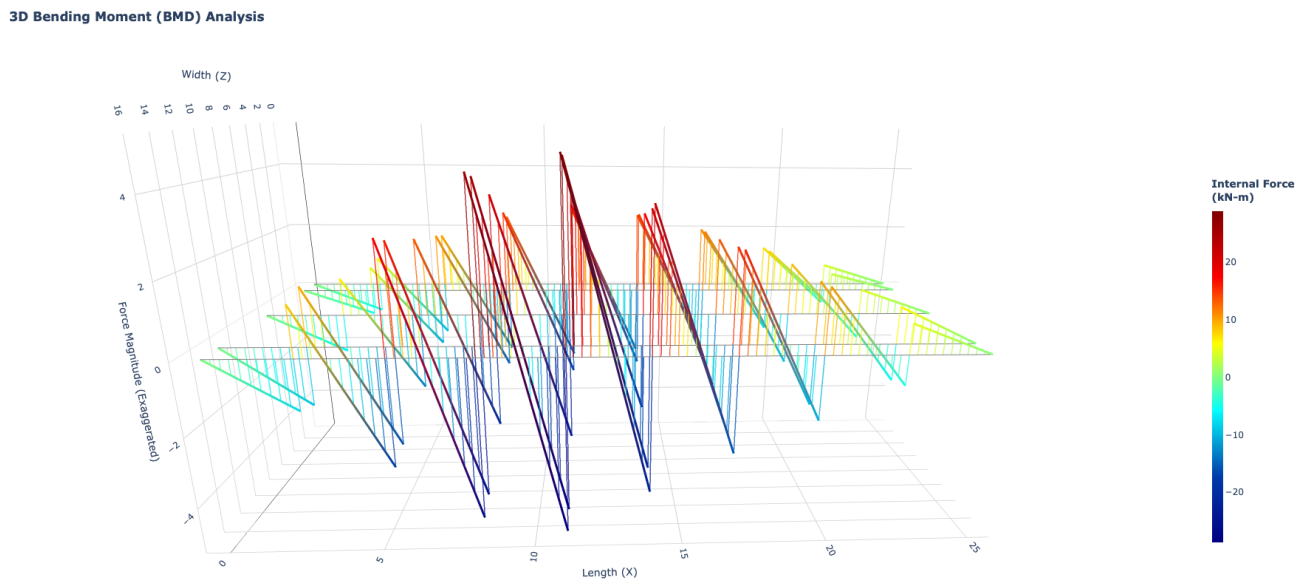

Figure 6: 3D Bending Moment Diagram (BMD)

5. Verification Against Marking Criteria

Correct node usage: node.py

Element connectivity: element.py

Sign convention: Preserved from dataset

Visual clarity: Scaling, axes, color mapping

MIDAS similarity: Vertical extrusion and heat-map coloring

6. Tools and Libraries Used

- Python 3.11

- Xarray

- NumPy

- Plotly

7. Conclusion

This project successfully demonstrates how internal force post-processing commonly performed in commercial software can be replicated using open-source Python tools.

8. Files Included

- xarray_usage.py

- task1_2d.py

- task2_3d.py

- node.py, element.py, screening_task.nc