

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:

- $V_y_i, V_y_j \rightarrow$ Shear force at start and end nodes
- $M_z_i, M_z_j \rightarrow$ 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)

The image shows two code editors side-by-side. The left editor is titled 'node.py' and contains Python code defining a dictionary of node coordinates. The right editor is titled 'element.py' and contains Python code defining a dictionary of element members.

```
node.py content:
1 # Node coordinates: node_id : [x, y, z]
2 nodes = {
3     1: [0.0000, 0.0000, 0.0000],
4     2: [0.0000, 0.0000, 1.2000],
5     3: [0.0000, 0.0000, 5.1750],
6     4: [0.0000, 0.0000, 9.1500],
7     5: [0.0000, 0.0000, 10.3500],
8     6: [25.0000, 0.0000, 0.0000],
9     7: [25.0000, 0.0000, 1.2000],
10    8: [25.0000, 0.0000, 5.1750],
11    9: [25.0000, 0.0000, 9.1500],
12    10: [25.0000, 0.0000, 10.3500],
13    11: [2.7778, 0.0000, 0.0000],
14    12: [2.7778, 0.0000, 1.2000],
15    13: [2.7778, 0.0000, 5.1750],
16    14: [2.7778, 0.0000, 9.1500],
17    15: [2.7778, 0.0000, 10.3500],
18    16: [5.5556, 0.0000, 0.0000],
19    17: [5.5556, 0.0000, 1.2000],
20    18: [5.5556, 0.0000, 5.1750],
21    19: [5.5556, 0.0000, 9.1500],
22    20: [5.5556, 0.0000, 10.3500],
23    21: [8.3333, 0.0000, 0.0000],
24    22: [8.3333, 0.0000, 1.2000],
25    23: [8.3333, 0.0000, 5.1750],
26    24: [8.3333, 0.0000, 9.1500],
27    25: [8.3333, 0.0000, 10.3500],
28    26: [11.1111, 0.0000, 0.0000],
29    27: [11.1111, 0.0000, 1.2000],
30    28: [11.1111, 0.0000, 5.1750],
31    29: [11.1111, 0.0000, 9.1500],
32    30: [11.1111, 0.0000, 10.3500],
33    31: [13.8889, 0.0000, 0.0000],
34    32: [13.8889, 0.0000, 1.2000],
```

```
element.py content:
1 # member_id : [start_node_id, end_node_id]
2 members = {
3     15: [3, 13],
4     24: [13, 18],
5     33: [18, 23],
6     42: [23, 28],
7     51: [28, 33],
8     60: [33, 38], ⚡
9     69: [38, 43],
10    78: [43, 48],
11    83: [48, 8],
12    14: [2, 12],
13    23: [12, 17],
14    32: [17, 22],
15    41: [22, 27],
16    50: [27, 32],
17    59: [32, 37],
18    68: [37, 42],
19    77: [42, 47],
20    82: [47, 7],
21    16: [4, 14],
22    25: [14, 19],
23    34: [19, 24],
24    43: [24, 29],
25    52: [29, 34],
26    61: [34, 39],
27    70: [39, 44],
28    79: [44, 49],
29    84: [49, 9],
30    13: [1, 11],
31    22: [11, 16],
32    31: [16, 21],
33    40: [21, 26],
34    49: [26, 31], ⚡
```

Figure 2: Example of node coordinates and element connectivity

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

Central Longitudinal Girder:

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

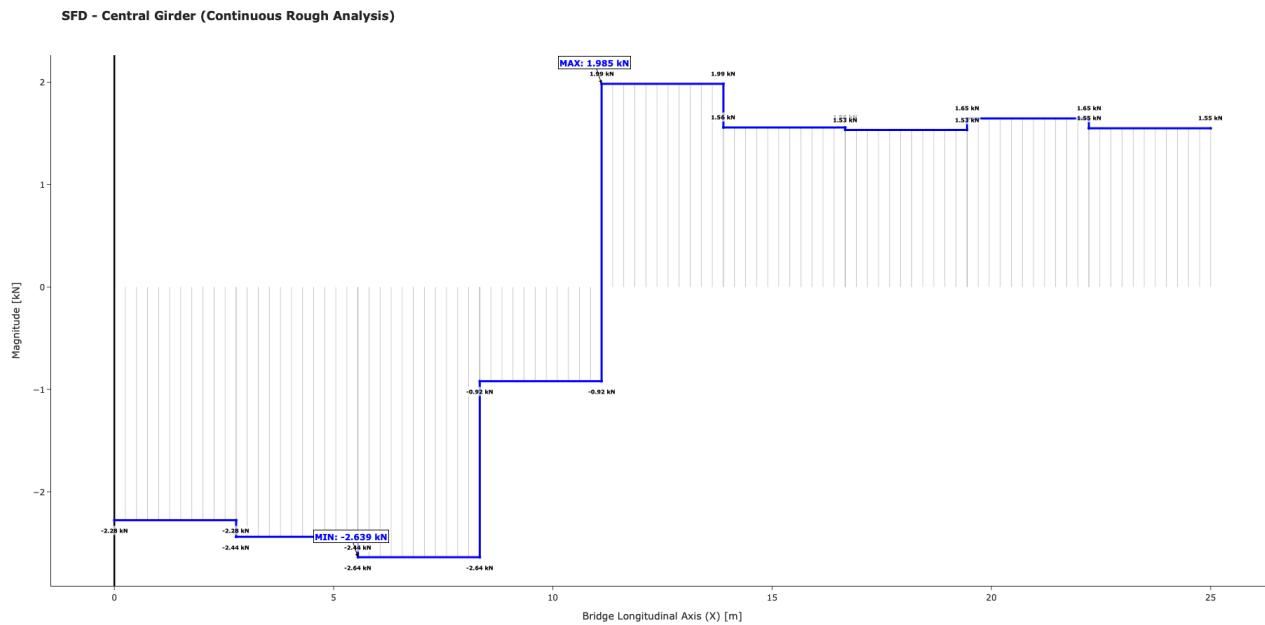


Figure 3: 2D Shear Force Diagram (SFD)

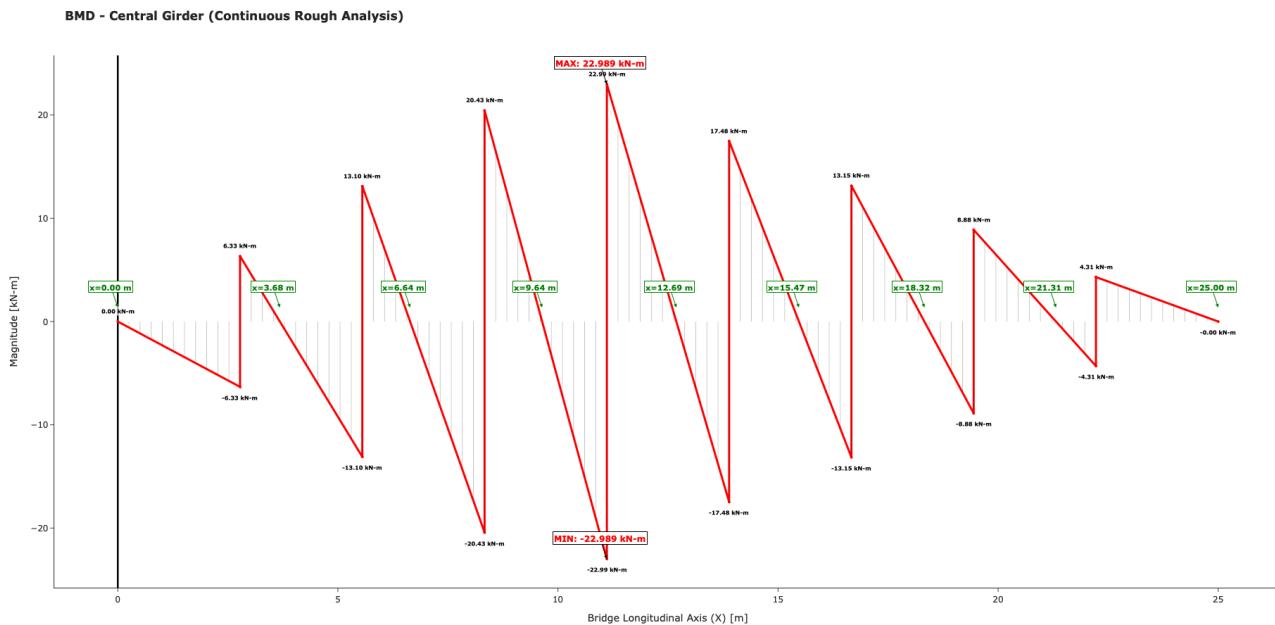


Figure 4: 2D Bending Moment Diagram (BMD)

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

Five girders visualized using exact node and element definitions.

3D Shear Force (SFD) Analysis

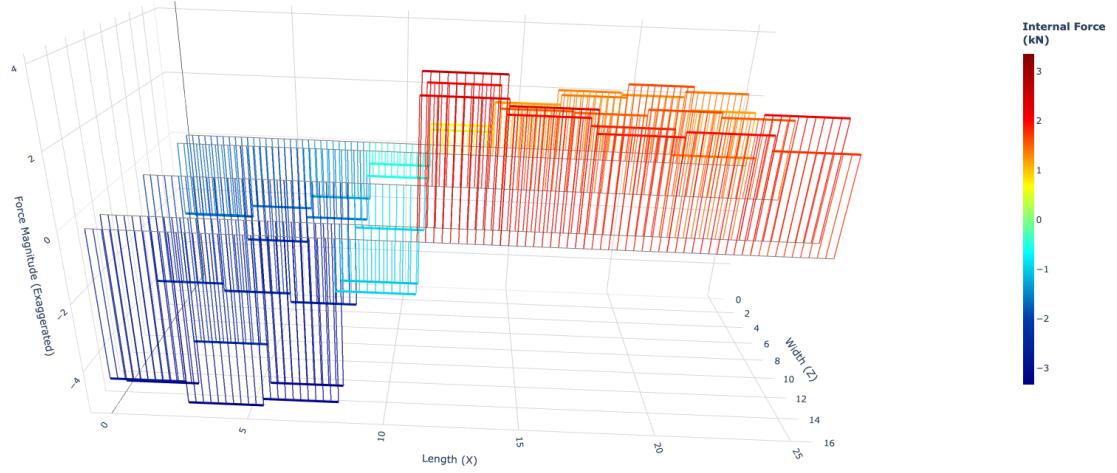


Figure 5: 3D Shear Force Diagram (SFD)

3D Bending Moment (BMD) Analysis

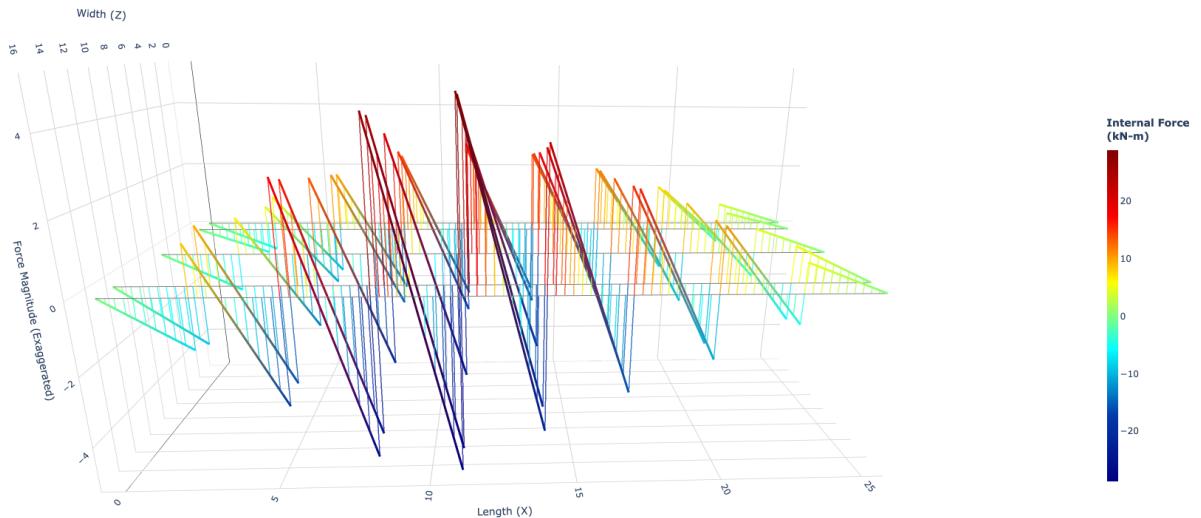


Figure 6: 3D Bending Moment Diagram (BMD)

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