Task Report

Xarray and Plotly/PyPlot

Name : Swetalina Sarangi
Registration Number : 24BCE10419
VIT Bhopal University

1. Introduction

This report documents the implementation and outcomes of the Osdag screening assignment involving **Xarray-based post-processing** and **2D/3D visualization of shear force and bending moment diagrams** using Plotly. The objective was to replicate industry-style structural post-processing for a steel bridge grillage model generated using **ospgrillage**.

The assignment consists of two compulsory tasks:

- **Task 1:** Generation of 2D Shear Force Diagram (SFD) and Bending Moment Diagram (BMD) for the **central longitudinal girder** using an Xarray dataset.
- **Task 2:** Generation of **3D SFD and BMD for all longitudinal girders**, visualized in a MIDAS-like style.

The solution is implemented in a modular and extensible manner, emphasizing correctness, clarity, and engineering consistency.

2. Dataset and Input Files

2.1 Xarray Dataset

The internal force and moment data is provided in NetCDF format (`Xarray_data.nc`) and loaded using the **xarray** library. Each element stores force components at its **i-end** and **j-end**. The relevant components used in this assignment are:

- **Mz**: Bending Moment
- **Vy**: Shear Force

The implementation automatically detects and standardizes the element dimension name to `element` to ensure robustness across datasets.

2.2 Geometry Definition

- **Node coordinates** are loaded from `node.py` as `{node_id: (x, y, z)}`.
- **Element connectivity** is loaded from `element.py` as `{element_id: (node_i, node_j)}`.

Standard finite element post-processing workflows are reflected in this division of geometry and force data.

3. Code Architecture

The codebase is organized into clear, reusable modules:

3.1 `load_data.py` – Data Loading Layer

This module defines the `DataLoader` class, which is responsible for:

- Loading and validating the Xarray dataset
- Importing node coordinates and element connectivity
- Extracting force components at the element level (`Mz_i`, `Mz_j`, `Vy_i`, `Vy_j`)

Key features:

- Automatic detection of element dimension name
- Flexible mapping of force component naming conventions
- Centralized accessors for geometry and force data

3.2 `geometry_utils.py` – Geometry and Force Processing

This module handles all girder-level preprocessing.

Girder Configuration

Five longitudinal girders are defined explicitly using element and node tags, as provided in the problem statement. Girder 3 is identified as the **central longitudinal girder**.

2D Force Extraction

For Task 1, force values are extracted along the girder length using:

- True **3D element length** computed from node coordinates
- Cumulative distance to ensure physically meaningful station locations

- Direct use of Xarray sign conventions (no manual sign flipping)

3D Geometry Extraction

For Task 2, nodal coordinates and corresponding force values are extracted for each girder to support 3D plotting.

A normalization-based scaling function is applied to force values to generate visually consistent diagram offsets without distorting relative magnitudes.

4. Task 1: 2D SFD and BMD for Central Girder

4.1 Objective

To generate accurate and continuous **2D Shear Force and Bending Moment diagrams** for the central longitudinal girder consisting of elements:

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

4.2 Methodology

- Extract `Mz_i`, `Mz_j`, `Vy_i`, and `Vy_j` from the Xarray dataset
- Compute cumulative girder length using actual 3D geometry
- Plot diagrams using **Plotly** with:
    - Continuous lines and markers
    - Zero reference lines
    - Filled regions for improved clarity
    - Engineering units and informative hover tooltips

4.3 Results

- The diagrams are continuous across element boundaries, confirming correct force mapping between adjacent elements.
- Position arrays for SFD and BMD are verified to be identical, ensuring consistency.
- Both static (PNG) and interactive (HTML) plots are generated.

5. Task 2 : 3D SFD and BMD for All Girders

5.1 Objective

To generate **3D shear force and bending moment diagrams** for all five girders, visually similar to MIDAS post-processing output.

5.2 Methodology

- Extract 3D geometry and force values for each girder
- Plot:
    - Base girder lines (structural framing)
    - Force/moment diagrams extruded in the **Y-direction** for visualization
    - Vertical connector lines between base geometry and force diagram

  Note: The Y-direction extrusion is purely for visualization and does **not** represent physical deformation.

5.3 Visualization Features

- Consistent global scaling across all girders
- Distinct color coding for each girder
- Interactive rotation, zooming, and hover inspection
- Proper axis labeling and aspect ratio control

5.4 Results

- Both 3D BMD and 3D SFD diagrams clearly display force variation along the bridge length.
- Relative force magnitudes between girders are visually comparable.
- The output closely matches the expected MIDAS-style interpretation.

6. main.py :

The file `main.py` serves as the **central driver script** for the entire OSDAG screening assignment. It orchestrates data loading, task execution, output generation, and error handling in a structured and user-friendly manner.

a) Project Path and Environment Setup

At the beginning of the script, project directories are defined using Python's `pathlib` module:

- `PROJECT_ROOT` identifies the root folder of the project

- `SRC_DIR`, `DATA_DIR`, and `PLOTS_DIR` correspond to the source code, input data, and output directories respectively

The `src/` directory is explicitly added to `sys.path` to ensure reliable module imports regardless of the execution environment.

b) Module Imports

After path setup, the script imports:

- `DataLoader` for handling all input datasets
- `create_2d_diagrams()` for Task 1 (2D SFD & BMD)
- `create_3d_diagrams()` for Task 2 (3D SFD & BMD)

This separation ensures modularity and clean responsibility boundaries between data handling and visualization logic.

c) Main Execution Flow

The `main()` function controls execution and performs the following steps sequentially:

1. **Project Structure Validation**
    - Confirms the presence of `src/` and `data/` directories
    - Creates the `plots/` directory if it does not already exist
2. **Data Initialization and Loading**
    - Instantiates the `DataLoader` class
    - Loads all required datasets using `loader.load_all()`
3. **Task 1 Execution – 2D SFD & BMD**
    - Generates bending moment and shear force diagrams for the central girder
    - Saves both static (`.png`) and interactive (`.html`) outputs
4. **Task 2 Execution – 3D SFD & BMD**
    - Generates 3D interactive diagrams for all girders
    - Outputs are saved in HTML format for interactive visualization
5. **Execution Summary**
    - Prints a clear list of all generated output files and their location

d) Error Handling and Robustness

The script includes structured exception handling to improve robustness:

- `FileNotFoundError` for missing input data or directories
- `ImportError` for module or path-related issues
- A generic `Exception` block that prints a full traceback for debugging

This design ensures that errors are reported clearly and execution fails gracefully when required.

7. How to Run, Test, and Access Outputs

7.1 Environment Setup

The codebase is written in Python and requires the following libraries:

- `xarray`
- `numpy`
- `plotly`
- `kaleido` (optional, for PNG export)

All required files should be placed in the prescribed directory structure, with the Xarray dataset and geometry files located inside the `data/` folder.

7.2 Running, Testing, and Main Script Execution

This project supports both **task-wise execution** and a **single consolidated execution using `main.py`**, which is the recommended approach for evaluators.

Step 1: Load and Verify Data

To verify that the dataset, node coordinates, and element connectivity are correctly loaded, run:

- `load_data.py`

This performs basic test calls such as:

- Accessing node coordinates
- Accessing element connectivity
- Extracting force components (`Mz_i`, `Mz_j`, `Vy_i`, `Vy_j`) for a sample element

Successful execution confirms that the input data is correctly interpreted.

Step 2: Task 1 – 2D SFD and BMD (Central Girder)

Run the following script:

- `task1.py`

This script:

- Extracts bending moment and shear force data for the central longitudinal girder
- Computes cumulative girder length using true 3D geometry
- Generates 2D SFD and BMD using Plotly

Outputs generated:

- `plots/task1_bmd.png` – Static Bending Moment Diagram
- `plots/task1_sfd.png` – Static Shear Force Diagram
- `plots/task1_bmd.html` – Interactive BMD
- `plots/task1_sfd.html` – Interactive SFD

Step 3: Task 2 – 3D SFD and BMD (All Girders)

Run the following script:

- `task2.py`

This script:

- Processes all five girders defined in the configuration
- Extracts nodal geometry and force values
- Generates MIDAS-style 3D SFD and BMD visualizations

Outputs generated:

- `plots/task2_3d_bmd.html` – Interactive 3D Bending Moment Diagram
- `plots/task2_3d_sfd.html` – Interactive 3D Shear Force Diagram

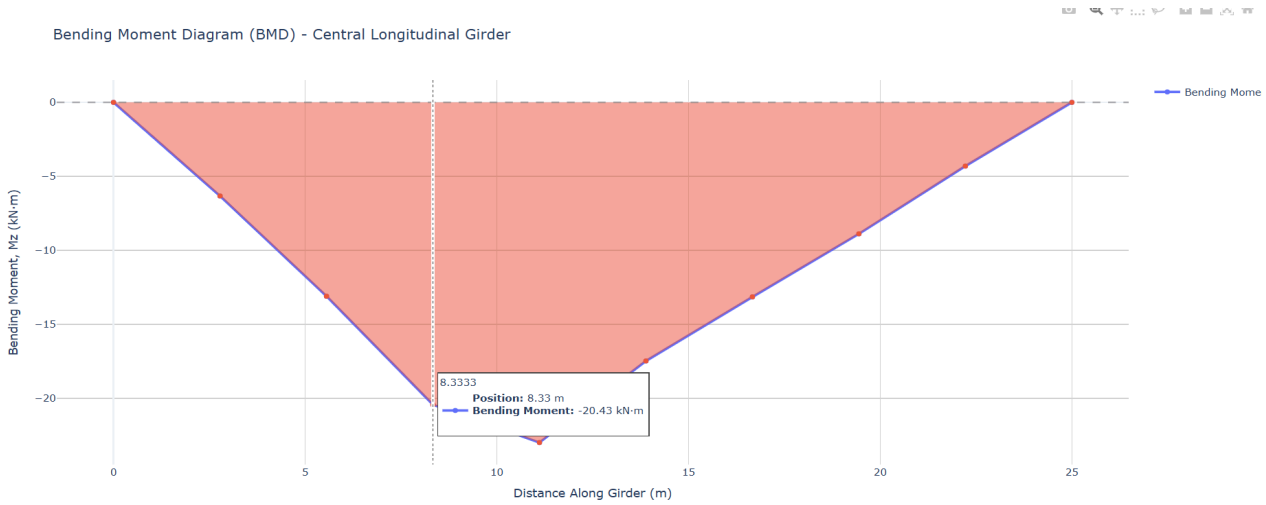
7.3 Accessing and Interpreting the Outputs

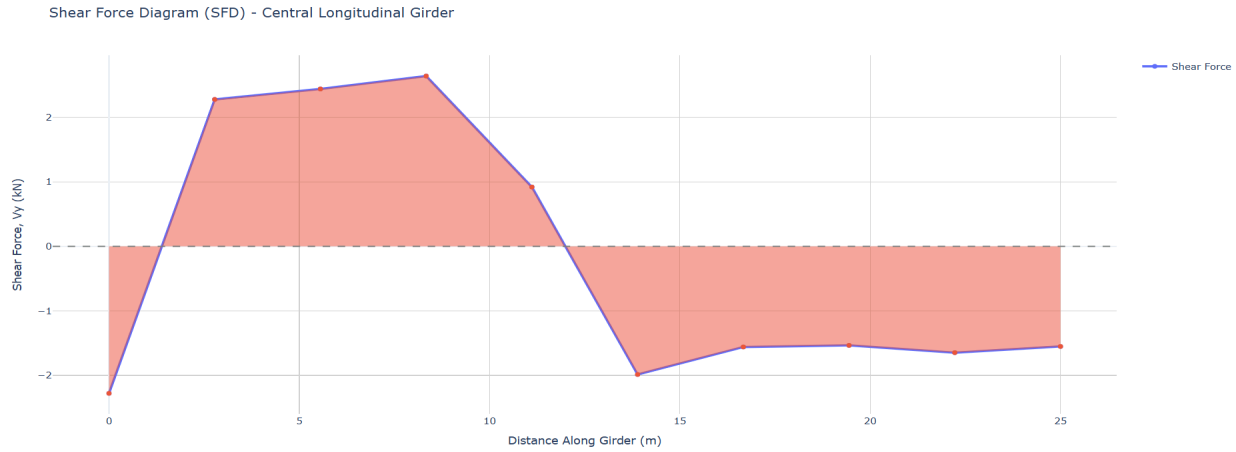
All output files are generated inside the `plots/` directory, regardless of whether the code is executed via `main.py` or individual task scripts.

- **PNG files** (`.png`) can be opened directly and are suitable for reports and documentation.
- **HTML files** (`.html`) can be opened in a web browser and enable :
    - Rotation and zooming of 3D plots
    - Examine force and moment values with a hover.
    - Clear identification of individual girders

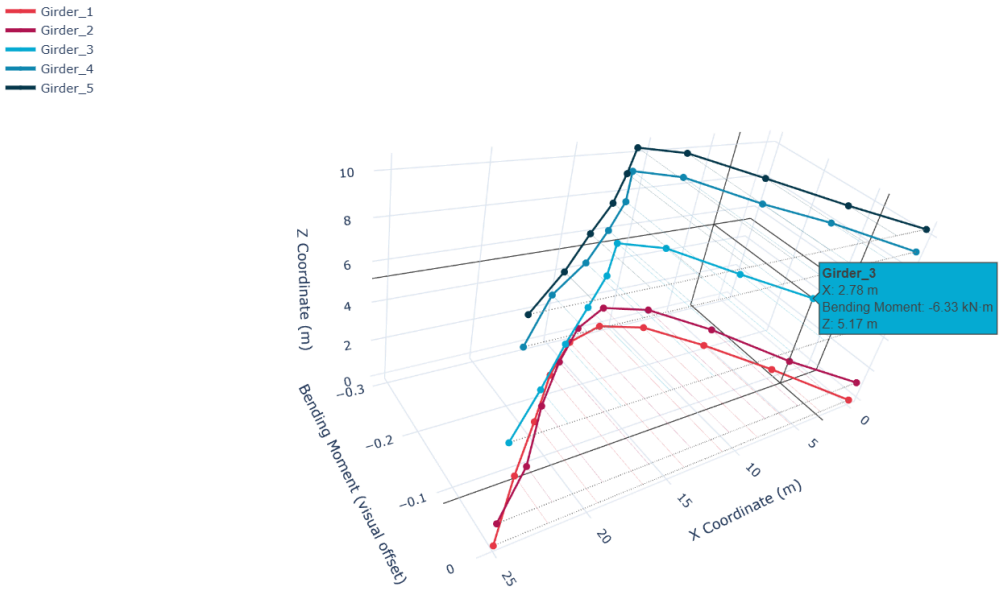
The `plots/` directory serves as the centralized output location for all generated plots.

Snapshots of the outputs :

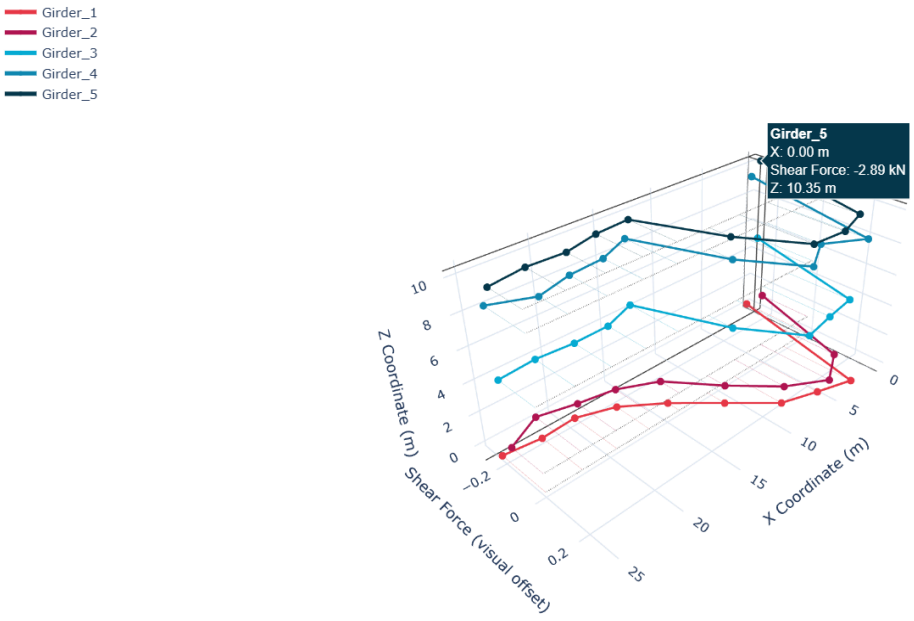




3D Bending Moment Diagram - All Girders (MIDAS Style)



3D Shear Force Diagram - All Girders (MIDAS Style)



8. Verification and Engineering Consistency

Several validation steps were included:

- Continuity of force values at element interfaces
- Range checks for bending moments and shear forces
- Geometry–force array length consistency
- Preservation of original Xarray sign conventions

These checks ensure that the plots are not only visually appealing but also structurally meaningful.

9. Conclusion

A clean, modular, and engineering-focused approach was used to successfully accomplish the assignment's goals.The implementation demonstrates:

- Correct and flexible use of Xarray for structural post-processing
- Precise extraction of shear force and bending moment data
- High-quality 2D and 3D visualizations aligned with professional structural software

This workflow is suitable for Osdag post-processing tasks in the real world because it can be directly extended to larger grillage models and additional force components.