Advanced Beam Analysis System Documentation

Engineering Team

April 9, 2025

Contents

1 Introduction

1.1 System Overview

The Advanced Beam Analysis System is a Python-based application that simulates and visualizes the behavior of a simply supported beam under moving loads. The system provides real-time analysis of:

- Shear force diagrams
- Bending moment diagrams
- Reaction forces at supports
- Envelope diagrams for maximum/minimum values

1.2 Key Features

- Interactive GUI with parameter controls
- Real-time animation of moving loads
- Calculation of maximum shear and moment values
- Visualization of force envelopes
- Dark mode interface with modern styling

2 Theoretical Background

2.1 Beam Analysis Fundamentals

The system analyzes a simply supported beam with two moving loads. The key equations used are:

2.1.1 Reaction Forces

For loads W_1 and W_2 at positions a and b respectively on a beam of length L:

$$R_A = W_1 \left(1 - \frac{a}{L}\right) + W_2 \left(1 - \frac{b}{L}\right)$$
$$R_B = W_1 \left(\frac{a}{L}\right) + W_2 \left(\frac{b}{L}\right)$$

2.1.2 Shear Force Calculation

The shear force $V(x)$ at any point x along the beam:

$$V(x) = \begin{cases} R_A & \text{if } x < a \\ R_A - W_1 & \text{if } a \leq x < b \\ R_A - W_1 - W_2 & \text{if } x \geq b \end{cases}$$

2.1.3 Bending Moment Calculation

The bending moment $M(x)$ at any point x:

$$M(x) = \begin{cases} R_A \cdot x & \text{if } x < a \\ R_A \cdot x - W_1 \cdot (x - a) & \text{if } a \leq x < b \\ R_A \cdot x - W_1 \cdot (x - a) - W_2 \cdot (x - b) & \text{if } x \geq b \end{cases}$$

3 System Implementation

3.1 Software Architecture

The application is built using:

- Python 3 with PyQt6 for the GUI
- Matplotlib for visualization
- NumPy for numerical calculations

3.2 Class Structure

The main components are:

- `BeamAnalysisApp`: Main application window
- `MplCanvas`: Custom matplotlib canvas for embedding plots

3.3 Key Components

3.3.1 User Interface

The GUI consists of:

- Input panel for parameters (beam length, loads, spacing)
- Control buttons (Start, Stop, Reset)
- Results display area
- Visualization canvas with multiple plots

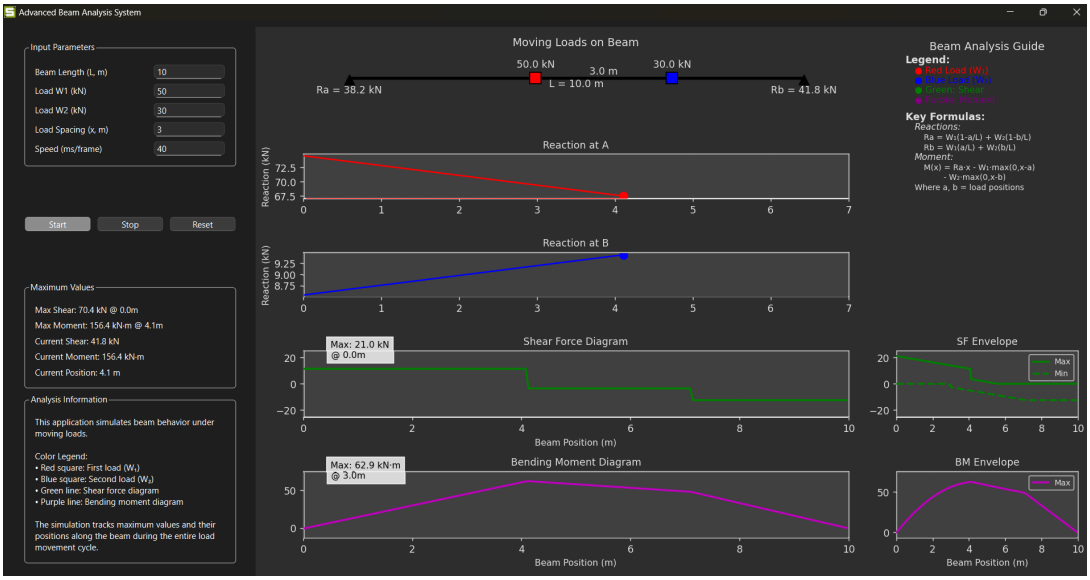

Figure 1: Application user interface layout

3.3.2 Visualization Components

The application displays six main plots:

1. Beam diagram with moving loads
2. Reaction force at support A
3. Reaction force at support B
4. Shear force diagram
5. Bending moment diagram
6. Envelope diagrams

3.4 Color Scheme

The application uses a consistent color palette:

- Beam loads: Red and blue squares
- Shear diagram: Green line
- Moment diagram: Purple line
- Maximum values: Highlighted with annotations

4 Installation and Execution

4.1 Prerequisites

Ensure you have Python 3.8+ installed with the following packages:

```
pip install pyqt6 matplotlib numpy
```

4.2 Running the Application

Execute the Python script:

```
python beam_analysis.py
```

5 Code Implementation

5.1 Main Application Class

The core functionality is implemented in the `BeamAnalysisApp` class:

```
class BeamAnalysisApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Advanced Beam Analysis System")
        self.setGeometry(100, 100, 1400, 1000)
        # ... initialization code ...

    def create_input_panel(self, main_layout):
        # ... GUI creation code ...

    def create_plots(self, main_layout):
        # ... Plot setup code ...

    def start_animation(self):
        # ... Animation control code ...
```

Listing 1: Main application class

5.2 Visualization Class

The `MplCanvas` class handles the plotting:

```
class MplCanvas(FigureCanvasQTAgg):
    def __init__(self, parent=None, width=12, height=10, dpi=100):
        self.fig = plt.figure(figsize=(width, height), dpi=dpi)
        gs = GridSpec(5, 2, figure=self.fig,
                      height_ratios=[1, 1, 1, 1.5, 1.5],
                      width_ratios=[3, 1])
        # ... axis creation code ...
```

Listing 2: Plotting canvas class

5.3 Animation Logic

The animation updates are handled by the `update` function:

```
def update(frame):
    current_pos = frame % (params['L'] - params['x'])
    a = current_pos
    b = current_pos + params['x']

    # Calculate reactions
    Ra = params['W1']*(1 - a/params['L']) + params['W2']*(1 - b/params['L'])
    Rb = params['W1']*(a/params['L']) + params['W2']*(b/params['L'])

    # Update visualizations
    self.update_beam_diagram(a, b, params['L'])
    self.update_reaction_plots(current_pos, Ra, Rb)
    self.update_diagrams(x_vals, shear, moment)
    self.update_envelopes()
```

Listing 3: Animation update function

6 Usage Instructions

6.1 Input Parameters

- **Beam Length (L)**: Total length of the beam in meters
- **Load W1**: Magnitude of first load in kN
- **Load W2**: Magnitude of second load in kN
- **Load Spacing (x)**: Distance between loads in meters
- **Speed**: Animation speed in milliseconds per frame

6.2 Controls

- **Start**: Begins the animation
- **Stop**: Pauses the animation
- **Reset**: Clears all plots and results

6.3 Interpreting Results

The application displays:

- Current and maximum shear forces
- Current and maximum bending moments
- Position of maximum values
- Envelope diagrams showing extreme values

7 Example Scenarios

7.1 Example 1: Equal Loads

For two 50 kN loads spaced 3 m apart on a 10 m beam:

- Maximum shear occurs when one load is at a support
- Maximum moment occurs when loads are centered

7.2 Example 2: Unequal Loads

For a 70 kN load followed by a 30 kN load:

- Maximum shear occurs when the larger load is at a support
- Maximum moment position depends on the load ratio

8 Troubleshooting

- **Animation not starting**: Check input values are valid numbers
- **Plots not updating**: Ensure speed parameter is reasonable (20-100 ms)
- **Loads disappearing**: Verify load spacing is less than beam length

9 Conclusion

The Advanced Beam Analysis System provides an interactive tool for visualizing beam behavior under moving loads. The system combines theoretical calculations with real-time visualization, making it valuable for engineering education and analysis.

A Complete Source Code

The full source code is available in the accompanying `beam_analysis.py` file.