The Journal of Open Source Education

BeamBending: a teaching aid for 1-D shear force and bending moment diagrams

Alfredo R. Carella[1]

1 OsloMet - Oslo Metropolitan University

DOI: 10.21105/jose.00060

Software

- Review
- Repository
- Archive

Submitted: 10 July 2019
Published: 29 July 2019

Summary

BeamBending is *both* an educational module *and* a Python package, intended to serve as a teaching aid during a first course in *Statics*. The aim of this module is to enhance clarity and provide visual hands-on examples while introducing the concepts of:

- stresses on slender *one-dimensional* solids (i.e. beams)
- normal force, shear force and bending moment diagrams

The package documentation includes a simple (but still rigorous enough) explanation of the background theory, inspired in (Beer, Russell Johnston, Dewolf, & Mazurek, 2017) and (Bell, 2015). It is assumed that the students understand static equilibrium of flat rigid bodies, but a short recap is provided. Code snippets that reproduce the theory examples are presented next to each result.

The package can be used by

- teachers who want to automatically create problem sets with their solutions (easily scriptable, *random-problem-generator friendly*);
- students who want to verify their solutions to introductory problem sets;
- students who like to play with example problems and receive immediate visual feedback (i.e. about how simple modifications to imposed loads affect the resulting reaction forces and internal stresses).

The `beambending` package is ready for installation using `pip`, or can be tested online using the provided Jupyter notebook.

Statement of Need

Statics courses in undergraduate engineering programs are sometimes taught before the knowledge of the relevant mathematical tools (i.e. simple calculus and linear vector algebra) is fully mature. Introducing a topic that resembles the mindset of calculus and employs a little intuitive standard sign convention, on top of a wobbly mathematical foundation, makes it fairly common for students to get lost in the calculations.

This package/module aims to bridge this gap and simplify students' first contact with this challenging new topic by working on two fronts simultaneously: 1. Explain the background theory from a simple example with focus on connecting the mathematical description with the physical beam model (`beambending` code snippets are interleaved in order to illustrate how the package works). 2. Provide a temporary scaffolding that helps to establish an immediate visual association between beam load states and internal stresses.

Functionality and Usage

A typical use case of the `beambending` package always involves creating an instance of the `Beam` class. The class constructor takes an optional *length* argument, which defaults to 10 in case no argument is provided.

```
from beambending import Beam
beam = Beam(9)  # Initialize a Beam object of length 9m
```

After a `Beam` object is created, the properties corresponding to the x-coordinates of the fixed and rolling supports must be defined.

```
beam.fixed_support = 2    # x-coordinate of the fixed support
beam.rolling_support = 7  # x-coordinate of the rolling support
```

Each load applied to the beam requires an instance of one of the load classes `DistributedLoadH`, `DistributedLoadV`, `PointLoadH`, or `PointLoadV`. The load classes are simply *namedtuples*, and make the resulting scripts easier to read by making the user's intention explicit. The symbolic variable `x`, also defined by the module, is used for defining variable distributed loads.

```
from beambending import DistributedLoadV, PointLoadH, PointLoadV, x
```

The loads can be applied to the `Beam` by passing an iterable (list or tuple) to the method `add_loads`.

```
beam.add_loads((
                PointLoadH(10, 3),  # 10kN pointing right, at x=3m
                PointLoadV(-20, 3),  # 20kN downwards, at x=3m
                DistributedLoadV(-10, (3, 9)),  # 10 kN/m, downward, for 3m <= x <
                DistributedLoadV(-20 + x**2, (0, 2)),  # variable load, for 0m <=
            ))
```

After the problem is fully defined (beam length + placement of supports + loads), the `plot` method can me invoked to plot a sketch of the loaded beam together with its corresponding load diagrams (normal force, shear force and bending moment).

```
fig = beam.plot()
```

The `plot` method is actually a wrapper that combines these four methods: `plot_beam_diagram`, `plot_normal_force`, `plot_shear_force` and `plot_bending_moment` into a single A4-sized printer-friendly plot.

Example

The following example, provided within the package documentation, summarizes the explanation above.

Input

```
from beambending import Beam, DistributedLoadV, PointLoadH, PointLoadV, x
beam = Beam(9)  # Initialize a Beam object of length 9m
beam.fixed_support = 2    # x-coordinate of the fixed support
beam.rolling_support = 7  # x-coordinate of the rolling support
beam.add_loads((
                PointLoadH(10, 3),  # 10kN pointing right, at x=3m
                PointLoadV(-20, 3),  # 20kN downwards, at x=3m
                DistributedLoadV(-10, (3, 9)),  # 10 kN/m, downwards, for 3m <= x
                DistributedLoadV(-20 + x**2, (0, 2)),  # variable load, for 0m <=
            ))
fig = beam.plot()
```

Output

The script above produces the following Matplotlib figure:

Recent Uses

I developed `beambending` for using it in my *Statics* course in the Autumn 2019 semester at OsloMet - Oslo Metropolitan University. Beta versions were tried and commented by students during the Autumn 2018 semester, but the effectiveness of the package has still not been tested with large groups of students.

References

Beer, F., Russell Johnston, E., Dewolf, J. T., & Mazurek, D. (2017). *Statics and Mechanics of Materials* (Second Edition.). McGraw-Hill Education.

Bell, K. (2015). *Konstruksjonsmekanikk - Del I: Likevektslære*. Fagbokforlaget.

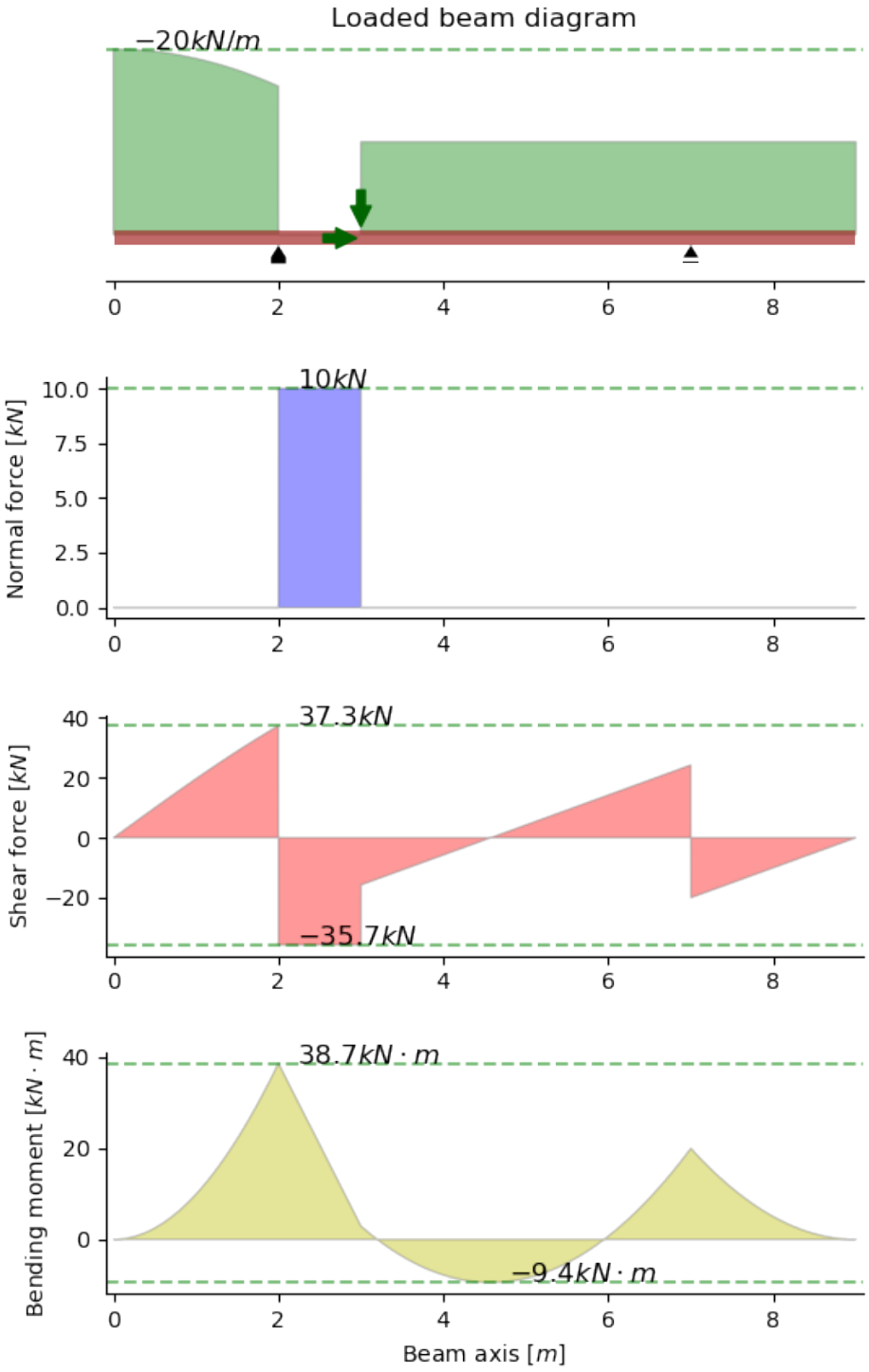

Figure 1: Output corresponding to the example code above