IndeterminateBeam: A Python package for solving 1D indeterminate beams

Jesse Bonanno

04 March 2021

Summary

`IndeterminateBeam` is a Python package aiming to serve as a foundation for civil and structural engineering projects in Python. The package can also serve as a standalone program and is useful for determining:

- reaction forces for indeterminate beams
- internal forces for indeterminate beams (shear, bending, axial)
- deflection of the beam due to resulting forces
- axial force, shear force, and bending moment diagrams

The package is based mainly on engineering concepts of statics as described in (Hibbeler, 2013), and Python packages Sympy (Meurer et al., 2017) and Plotly (Plotly Technologies Inc, 2015). The package documentation provides a brief overview of the theory behind the solutions used to calculate the forces on the indeterminate beam.

The `IndeterminateBeam` package repository can be found on Github and is ready for installation using `pip`. A text-based example of the package can be found on this Jupyter Notebook and a web-based graphical user interface (GUI) is available at https://indeterminate-beam.herokuapp.com/.

Project Purpose

The purpose of this project was two-fold:

1. Create a free website that has more features than paid and free beam calculators that exist on the web.
2. Provide a foundation for civil and structural engineers who want to create higher order engineering Python programs.

Several (mostly paid) beam calculator websites currently exist online, providing the same service as this package, with web traffic in the hundreds of thousands per month (Similiarweb, 2021). Despite this, no online service exists (to the authors knowledge) that has all the features of `IndeterminateBeam` and is also free.

Similiarly, there are no well-documented indeterminate beam solving Python packages (to the authors knowledge) despite the importance of such a calculation in engineering. Several python finite element analysis (FEA) packages do exist, however they are vastly overcomplicated for someone wanting to only solve for forces on a one-dimensional beam.

This python package was heavily inspired by beambending (Carella, 2019), a module created for educational purposes by Alfredo Carella of the Oslo Metropolitan University. The beambending module, although well documented, can only solve for simply supported beams consisting of a pin and roller support. The package documentation for this project includes a more rigorous overview of the theory behind the basics for solving determinate structures.

Project Comparison

A feature comparison in Table 1 below has been taken from beambending (Carella, 2019) and modified to include more packages and features.

	Arbitrary distributed load functions	Full GUI (no code required)	Arbitrary number of loads	Free	Open Source	Featured theory module	Detailed solution procedure	Programmable interface	Spring Supports	Any DOF combination for Supports	Any number of supports
IndeterminateBeam	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
BeamBending	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗
Beam Calculator Online	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
Beam Guru	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓
MechaniCalc	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✓
SkyCiv Beam	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
Steel Beam Calculator	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Structural Beam Calculator	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗
WebStructural	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓
ClearCalcs	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓

Table 1: Summary of feature comparison with existing packages

There are five main differences between the `IndeterminateBeam` package and the rest of the reviewed tools:

- Arbitrary distributed load functions are accepted (as long as Sympy can parse them), i.e. they are not restricted to constants or linear functions.
- The package can be called from regular Python code, which makes it easy to implement higher order engineering solutions.
- It is not only free but also completely open-source.
- Spring supports can be modelled
- Any degree of freedom combination can be constructed for supports

Shortfalls

The solution is completely analytical and does not use finite element analysis. This makes the analysis slightly slower than if it were complete using finite element analysis, however the benefit trade off is code clarity, which was a primary objective of this project.

The website is hosted on Heroku free of cost. This affects the website in two ways:

1. The website sleeps after 30 minutes of inactivity and may then take up to 20 seconds to load.
2. The server can occassionally be slow.

Functionality and Usage

A typical use case of the `IndeterminateBeam` package involves the following steps:

1. Create a `Beam` object
2. Create `Support` objects and assign to `Beam`
3. Create `Load` objects and assign to `Beam`
4. Solve for forces on `Beam` object
5. Plot results

You can follow along with the example below in this web-based Jupyter Notebook.Units and load direction conventions are described in the package documentation.

Creating a Beam

The creation of a `Beam` instance involves the input of the beam length (m) and optionally the input of the Young's Modulus (E), second moment of area (I), and cross-sectional area (A). E, I and A are optional and by default are the properties of a steel 150UB18.0. For a beam with constant properties, these parameters will only affect the deflections calculated and not the distribution of forces, unless spring supports are specified.

```
from indeterminatebeam import Beam
# Create 7 m beam with E, I, A as defaults
beam = Beam(7)
# Create 9 m beam with E, I, and A assigned by user
beam_2 = Beam(9,E=2000, I =10**6, A = 3000)
```

Defining Supports

`Support` objects are created separately from the `Beam` object, and are defined by an x-coordinate (m) and the beams translational and rotational degrees of freedom.

Degrees of freedom are represented by a tuple of 3 booleans, representing the x , y , and m directions respectively. A `1` indicates the support is fixed in a direction and a `0` indicates it is free.

Optionally, stiffness can be specified in either of the translational directions, which overrides the boolean specified.

```
from indeterminatebeam import Support
a = Support(5,(1,1,0))      # Defines a pin support at location x = 5m
b = Support(0,(0,1,0))      # Defines a roller support at location x = 0m
c = Support(7,(1,1,1))      # Defines a fixed support at location x = 7m
# Assign the support objects to a beam object created earlier
beam.add_supports(a,b,c)
```

Defining loads

`Load` objects are created separately from the `Beam` object, and are generally defined by a force value and then a coordinate value, however this varies slightly for different types of loading classes.

```
from indeterminatebeam import PointLoadV, PointTorque, DistributedLoadV
load_1 = PointLoadV(1,2)                # Create 1kN point load at x = 2m
load_2 = DistributedLoadV(2,(1,4))      # Create a 2kN UDL from x = 1m to x = 4m
load_3 = PointTorque(2, 3.5)            # Defines a 2kN.m point torque at x = 3.5m
beam.add_loads(load_1,load_2,load_3)    # Assign the load objects to the beam object
```

Solving for Forces

Once the `Beam` object has been assigned with `Load` and `Support` objects it can then be solved. To solve for reactions and internal forces we call the analyse function.

```
beam.analyse()
```

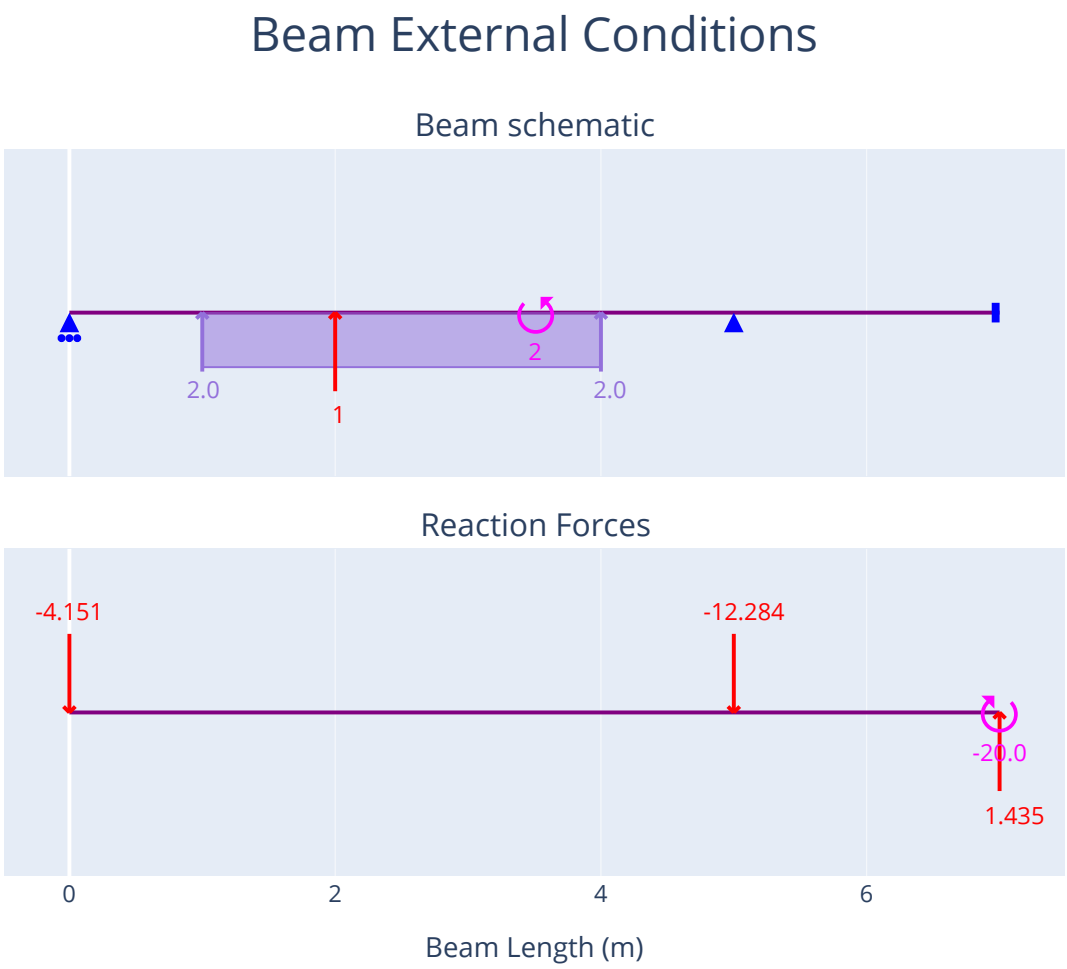
Plotting results

After the beam has been analysed we can plot the results.

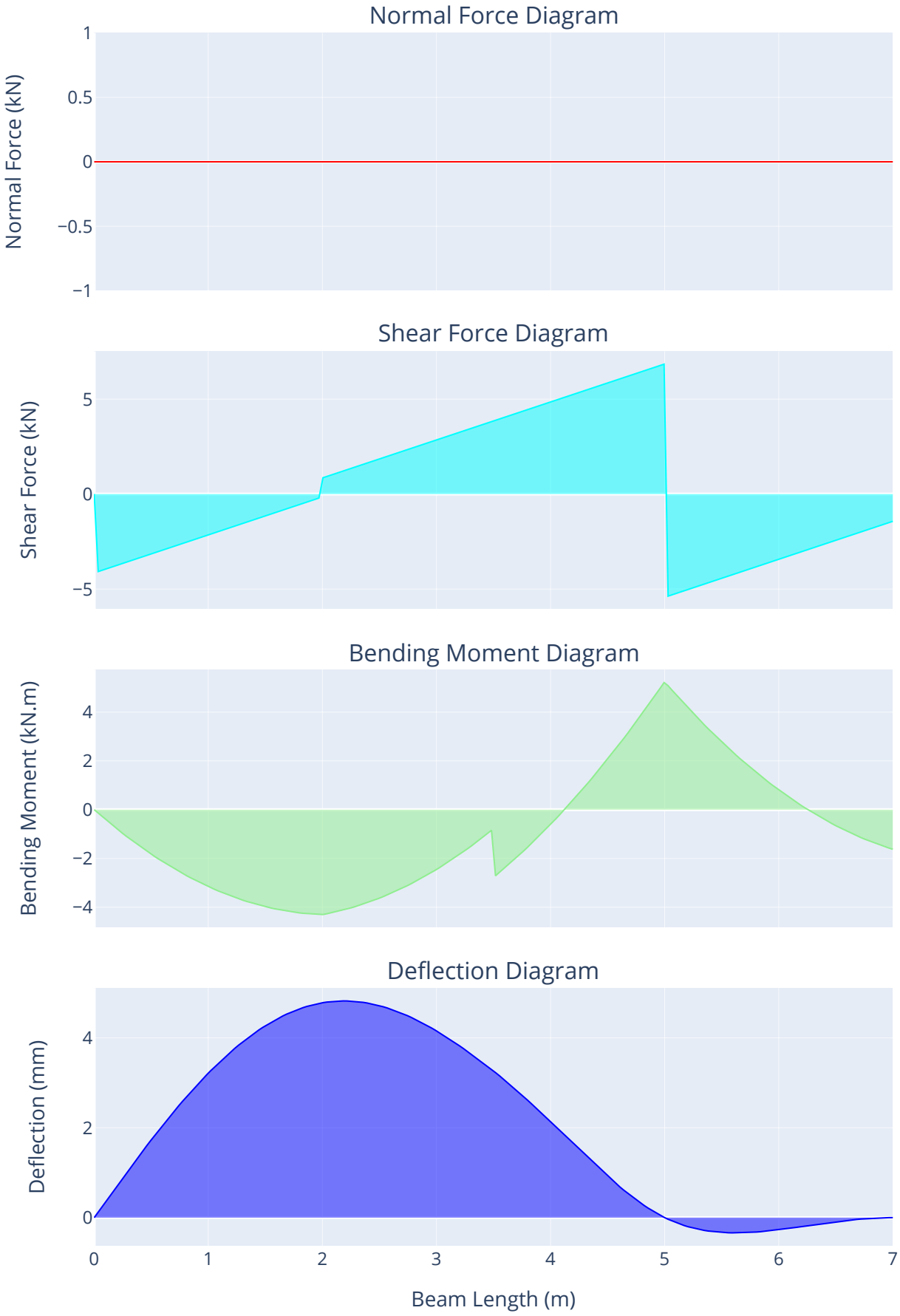
```
beam.plot_beam_external()
beam.plot_beam_internal()
```

The `plot_beam_external` and `plot_beam_internal` methods collate otherwise seperate plots.

The script above produces the following figures:



Analysis Results



References

1. *Beam Calculator Online*. (2020). http://rascheta.net/beamuk/; Accessed on 2020-12-21.
2. *Beam Guru*. (2020). http://beamguru.com; Accessed on 2020-12-21.
3. Carella, A. (2019). BeamBending: A teaching aid for 1-d shear force and bending moment diagrams. *Journal of Open Source Education, 2*(16), 65. https://doi.org/10.21105/jose.00065
4. Hibbeler, R. (2013). *Mechanics of materials*. P.Ed Australia. ISBN: 9810694369
5. *MechaniCalc*. (2019). https://mechanicalc.com/calculators/beam-analysis/; Accessed on 2020-12-21.
6. Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., ... Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science, 3*, e103. https://doi.org/10.7717/peerj-cs.103
7. *Similiarweb*, 2021, https://www.similarweb.com/. Accessed 1 Mar 2021.
8. *SkyCiv Beam*. (2017). https://skyciv.com/structural-software/beam-analysis-software/; Accessed on 2020-12-21.
9. *Steel Beam Calculator*. (2020). https://www.steelbeamcalculator.com/; Accessed on 2020-12-21.
10. *Structural Beam Deflection and Stress Calculators*. (2020). https://www.amesweb.info/StructuralBeamDeflection/BeamDeflectionCalculators.aspx; Accessed on 2020-12-21.
11. Plotly Technologies Inc. *Collaborative data science*. Montréal, QC, 2015. https://plot.ly."
12. *WebStructural*. (2020). https://webstructural.com/; Accessed on 2020-12-21.

Appendix

Project Components

I have created the entire project with Python.

To be more specific the project used:

- Sympy and Numpy for calculations
- Plotly for plotting
- Dash.plotly for website back-end
- Heroku for website hosting
- Sphinx and Markdown for documentation
- Sphinx-Bibtex and Bibtex for referencing
- PyPi for package distribution
- Jupyter Notebooks for example uses
- Google Colab and Binder for hosting jupyter notebooks
- Read the Docs for documentation hosting
- Github for version control
- Unittest and Pytest for unit testing
- Travis CI for continuous integration

Licensing and Contributing

Acknowledgments

Thanks to:

- Alfredo Carella, whose Python package beambending inspired this project, and for helpful advice regarding the implementation of the Sympy module.
- Andrew Rong and Michael Hou for website testing.
- Cat Wong for editorial review.