beambending Documentation

Release 1.0

Alfredo Carella

May 04, 2020

Getting started

Okay, which concerns does this project address?

- You are taking your first course on statics and you would like to have a more interactive way of visualizing shear forces and bending moments diagrams.
- You feel you understand better when you have some examples you can modify at will.
- You want to review the basic theory, double-check that you are doing things right and test yourself with some new problems.
- You are teaching a course on statics and want to generate practice problems automatically.

If any of these statements describe you, it may be a good idea to take a quick look at this project.

CHAPTER 1

First steps

- The easiest way to try this package is using a web-based notebook:

- You can also download and install the `beambending` package, which runs on Python 3.
  - If you do not have a Python 3 interpreter on your machine, you can install the last version following the instructions in this tutorial.
  - Once you have Python 3, you can open a terminal and install the package with this one-liner:

    ```
    python3 -m pip install --user beambending
    ```

    NOTE: You may need to replace `python3` above by the path to your Python 3 executable, or simply `python` if you are running Windows.

Tip: If you are not 100% familiar with basic one-dimensional elasticity (i.e. shear force and bending moment diagrams), I recommend you to start by going through the section *Background knowledge*.

This project is completely open source, and the code can be found in this GitHub repository.

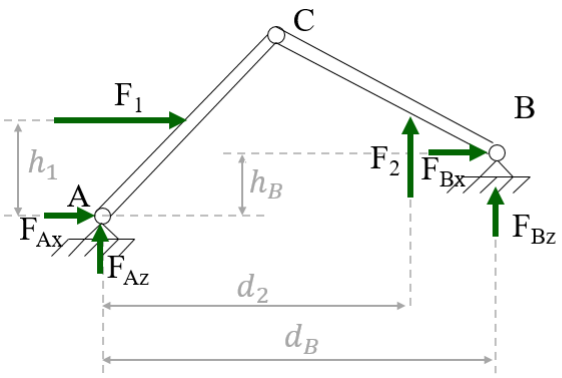
CHAPTER 2

Background knowledge

In this page we will go through introductory bending theory, as typically covered in an introductory statics course. We will start by a quick overview of the required knowledge, and then derive our analysis from basic principles.

Our starting point will be the fact that for a body to be at rest, the **vector sum** of all external forces $\mathbf{F_i}$ and moments $\mathbf{M_i}$ acting on it must be zero.

$$\sum \mathbf{F_i} = 0; \quad \sum \mathbf{M_i} = 0 \tag{2.1}$$



$$F_{Ax} + F_{Bx} + F_1 = 0$$
$$F_{Az} + F_{Bz} + F_2 = 0$$
$$F_2 d_2 + F_{Bz} d_B + F_{Bx} h_B - F_1 h_1 = 0$$

Fig. 1: A rigid body in equilibrium (i.e. whose sum of both forces and moments equal zero)

Furthermore, for a system to be at rest, each of its components need to be at rest. This means that Eq. (2.1) must be satisfied **for each** component in our system.

Note that $\mathbf{F_{1C}} = -\mathbf{F_{2C}}$, according to the *action and reaction principle*. If you are still not 100% comfortable with the action and reaction principle, you should review that before proceeding. As a self-test, the concept presented in this educational video should be completely obvious to you.

So far, we have only thought about reaction forces as applied *externally* to one or more rigid bodies. In other words, each rigid body has been considered to be fully contained in a single subsystem. However, Eq. (2.1) can tell us much

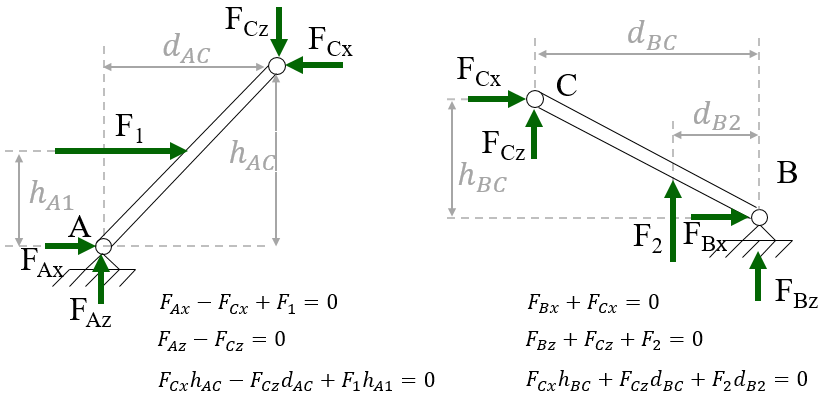


Fig. 2: For a system to be in equilibrium, each of its subsystems **must** be in equilibrium.

more than that if we lift that restriction. This equation applies to every arbitrary subset of a body, and not only to full bodies. We are going to exploit this to a larger extent in the next section.

CHAPTER 3

Beam diagrams

To state it explicitly once more, the calculation of internal forces for mechanical systems at rest will be based on:

1. The necessary and sufficient conditions for static equilibrium of a system,
2. The fact that any arbitrary part of a system is on itself a new system.

So... what if we chose some of the imaginary boundaries that split our system so that they *cut through* a body? Let's consider a system consisting of a single *beam*[1] and analyze what happens internally to this body.

3.1 Normal force $N(x)$

We reserve the name of *normal forces* for those who are perpendicular to a given plane of interest. In the context of beams (i.e. slender components), we subdivide a body into subsystems by drawing planes perpendicular to its main axis (*x-axis*), as shown in the figure below. Normal forces are hence perpendicular to these planes, i.e. parallel to the beam's main axis.

The first introductory problem to consider will be a 6 meter long horizontal beam resting on two supports:

- a roller support at $x = 0$
- a pinned support at $x = 3$ m

Show/Hide code for generating the image below

```
beam = Beam(6)
beam.rolling_support = 0  # x-coordinate of the rolling support
beam.pinned_support = 3    # x-coordinate of the pinned support
beam.add_loads([PointLoadH(5, 6)])  # 5 kN pointing right, at x=6 m
fig = beam.plot_beam_diagram()
ax = fig.gca()
ax.text(0, -1.35, r'A', ha='center', va='top')
ax.text(3, -1.35, r'B', ha='center', va='top')
```

(continues on next page)

[1] A *beam* will be defined in this context as a long, slender component (i.e. whose length along its main axis is significantly larger than on directions perpendicular to it).

(continued from previous page)

```
ax.text(6, -0.5, r'$F_1=5$ kN', ha='right', va='top', color='darkgreen')
ax.axvline(x=2, linestyle='--', color="k", alpha=0.5)
ax.text(2, 0.5, r'Section 1-1', rotation=90, ha='right', va='bottom')
ax.axvline(x=4, linestyle='--', color="k", alpha=0.5)
ax.text(4, 0.5, r'Section 2-2', rotation=90, ha='right', va='bottom')
ax.axvline(x=6, linestyle='--', color="k", alpha=0.5)
ax.text(6, 0.5, r'Section 3-3', rotation=90, ha='right', va='bottom')
```

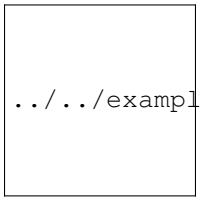


Fig. 1: The beam is held in place by a rolling support at *A* and a pinned support at *B*. A normal (tensile) force $F_1 = 5$ kN is acting at the right end of the beam.

Let's find the reaction forces first, so we can proceed with our analysis.

$$\sum F_x = 0 \implies F_{Bx} + F_1 = 0 \implies \underline{F_{Bx} = -5\ \text{kN}}$$

$$\left.\begin{array}{l} \sum M_A = 0 \\ \sum F_z = 0 \end{array}\right\} \implies \underline{F_A = F_{Bz}} = 0$$

Note: To keep matters simple, the weight of the beam is neglected throughout this analysis.

With the information about the reaction forces at supports *A* and *B*, let's see what happens to the subsystem between the plane 2-2 (located between support *B* and the right end of the beam) and the plane 3-3 (exactly at the right end of the beam, where the load $\mathbf{F_1}$ is acting).

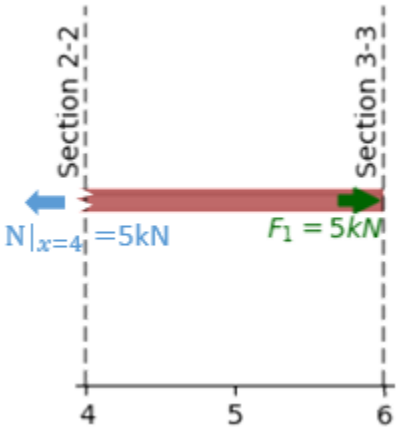


Fig. 2: Subsystem between planes 2-2 and 3-3, to the right of support *B*.

Let's apply the equilibrium equations to this subsystem. Since we only have *colinear* forces along the x-axis (we neglect the height of the beam), we write the sum of forces along this direction. The other equilibrium equations are identically zero (no vertical forces or moments acting on the beam), so they don't add any new information.

$$\sum F_x = 0 \implies F_1 - N = 0 \implies \underline{N|_{x=x_2} = 5\ \text{kN}}$$

Here we have found the internal normal force $\mathbf{N}$ to be equal to 5 kN. The positive sign corresponds to the standard convention. The normal force $\mathbf{N}$ is defined as positive when it points outwards. It follows from this that tensile forces are positive and compression forces negative. The result is the same for any plane located between the support *B* (section 2-2) and the right end of the beam (section 3-3).

Next, let's consider a slightly different subsystem between planes 1-1 and 3-3 and perform the same analysis

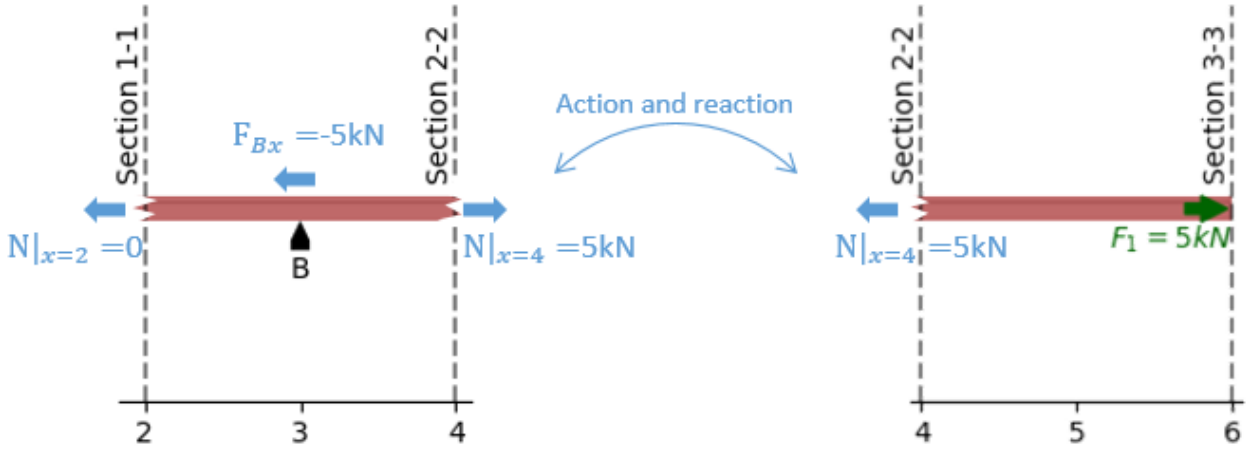


Fig. 3: Subsystem between planes 1-1 (btw supports *A* and *B*) and 3-3 (beam's right end)

$$\sum F_x = 0 \implies N + F_{Bx} - F_1 = 0 \implies \underline{N|_{x=x_1} = 0}$$

The normal force $\mathbf{N}$ across the section 1-1 is zero. Correspondingly, the result is the same for any plane located between supports A (section 1-1) and B (section 3-3).

If we join these last two results, we can reconstruct the normal force $\mathbf{N}$ along the full beam span as a function of the $\mathbf{x}$ coordinate, which we will predictably call $\mathbf{N}(\mathbf{x})$. In this simple case, we end up with the following piecewise function:

$$N(x) = \begin{cases} 0 & \text{if } 0 \leq x < x_B \\ 5 \text{ kN} & \text{if } < x_B \leq x < L \end{cases}$$

Show/Hide code for generating the image below

```
beam = Beam(6)
beam.rolling_support = 0  # x-coordinate of the rolling support
beam.pinned_support = 3    # x-coordinate of the pinned support
beam.add_loads([PointLoadH(5, 6)])  # 5 kN pointing right, at x=6 m
fig = beam.plot_normal_force()
```

../../examples/example_normal1.png

3.2 Shear force $V(x)$ and moment $M(x)$

Let's do a similar analysis of the same beam for vertical forces. Instead of the horizontal force $\mathbf{F_1}$, consider now a vertical force $\mathbf{P_1} = 10$ kN acting at the beam's right end (plane 3-3).

Show/Hide code for generating the image below

```
my_beam = Beam(6)
my_beam.rolling_support = 0  # x-coordinate of the rolling support
my_beam.pinned_support = 3    # x-coordinate of the pinned support
my_beam.add_loads([PointLoadV(10, 6)])  # 10 kN pointing upwards, at x=6 m
fig = my_beam.plot_beam_diagram()
ax = fig.gca()
ax.text(0, -1.35, r'A', ha='center', va='top')
ax.text(3, -1.35, r'B', ha='center', va='top')
ax.text(5.9, -1.2, r'$P_1=10$ kN', ha='right', va='top', color='darkgreen')
ax.axvline(x=2, linestyle='--', color="k", alpha=0.5)
ax.text(2, 0.5, r'Section 1-1', rotation=90, ha='right', va='bottom')
ax.axvline(x=4, linestyle='--', color="k", alpha=0.5)
ax.text(4, 0.5, r'Section 2-2', rotation=90, ha='right', va='bottom')
ax.axvline(x=6, linestyle='--', color="k", alpha=0.5)
ax.text(6, 0.5, r'Section 3-3', rotation=90, ha='right', va='bottom')
```

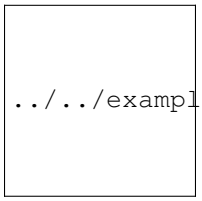


Fig. 4: The beam is held in place by a rolling support at *A* and a pinned support at *B*. A force $P_1 = 10$ kN directed upwards is acting at the right end of the beam.

In the same way as before, we start by finding the reaction forces at the supports *A* and *B*.

$$\sum F_x = 0 \implies \underline{F_{Bx} = 0}$$

$$\sum M_A = 0 \implies F_{Bz} = -\frac{\mathbf{P_1} L}{d} = -20 \text{ kN}$$

$$\sum F_z = 0 \implies F_A = \frac{\mathbf{P_1}(L-d)}{d} = 10 \text{ kN}$$

where $L = 6$ m is the length of the beam, and $d = 3$ m is the distance between supports *A* and *B*.

Next, we draw a free body diagram of the beam section comprised between planes 2-2 and 3-3, and do a balance of forces and moments once more.

$$\sum F_z = 0 \implies \mathbf{P_1} - V = 0 \implies V = 10 \text{ kN}$$

$$\sum M = 0 \implies M(x) - \mathbf{P_1}(x - L) = 0 \implies M(x) = \mathbf{P_1}(x - L)$$

The vertical plane 2-2, corresponds to $x = 4$ m, hence

- $V|_{2-2} = V(x)|_{x=4\text{ m}} = \underline{10 \text{ kN}}$, and
- $M|_{2-2} = M(x)|_{x=4\text{ m}} = \underline{-20 \text{ kN·m}}$.

The negative value of $M(x)|_{x=4\text{ m}}$ means that the actual moment acting on the left side of that beam section is *opposite to the arrow drawn in the figure above* (i.e. clockwise instead of counterclockwise).

Note: The sign convention we follow is shown in the figure below:

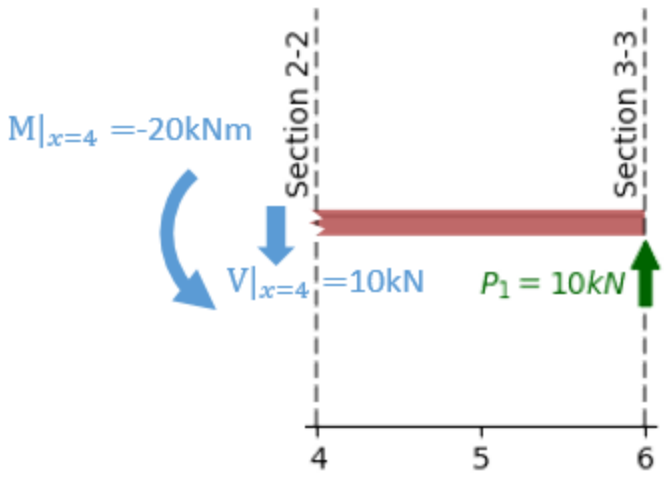


Fig. 5: Subsystem between planes 2-2 and 3-3, to the right of support *B*.

- A positive normal force *N* contributes to stretching (elongating) of the beam.
- A positive shear force *V* points upwards (z+) at the right side of the beam (x+), and downwards (z-) at the left side of the beam (x-).
- A positive bending moment causes a contributes to stretch on the upper side of the beam and compression on its lower side.

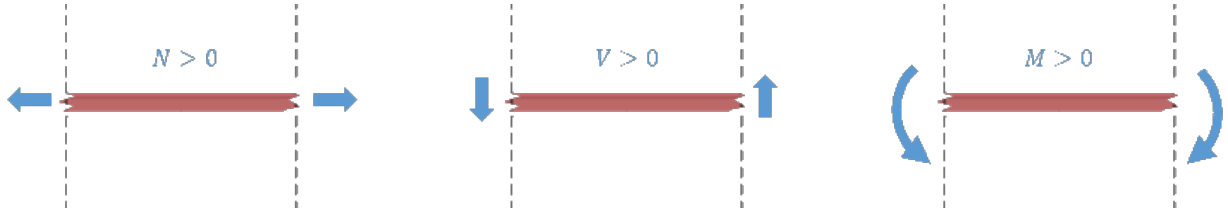


Fig. 6: Sign convention used for the beam diagrams: the arrow directions correspond to positive values.

Let's calculate now the shear force and bending moment at the vertical plane 1-1. To that end, we will consider the beam section between planes 1-1 and 2-2, as shown in the figure below.

The equations for sum of forces and sum of moments become then:

$$\begin{aligned} \sum \mathbf{F_z} = -V_{1-1} + F_{Bz} + V_{2-2} &= 0 \\ -V_{1-1} + (-20\ \text{kN}) + 10\ \text{kN} &= 0 \\ V_{1-1} &= \underline{-10\ \text{kN}} \end{aligned}$$

$$\begin{aligned} \sum \mathbf{M} = M_{1-1} + F_{Bz}(d-x) - V_{2-2}(4\ \text{m} - x) &= 0 \\ M_{1-1} + 20\ \text{kN}(3\ \text{m} - x) - 10\ \text{kN}(4\ \text{m} - x) &= 0 \\ \text{since } x|_{1-1} = 2\ \text{m} \implies M_{1-1} + 20\ \text{kN}(3\ \text{m} - 2\ \text{m}) - 10\ \text{kN}(6\ \text{m} - 2\ \text{m}) &= 0 \\ M_{1-1} &= \underline{-20\ \text{kN·m}} \end{aligned}$$

Tip: This result would (of course) have been the same if our free-body diagram had included the whole beam to the right of the plane 1-1. The same is true for a free-body of the left side of the beam. In order to make sure you will understand the next section, I suggest you **stop reading for a moment and try to verify this**.

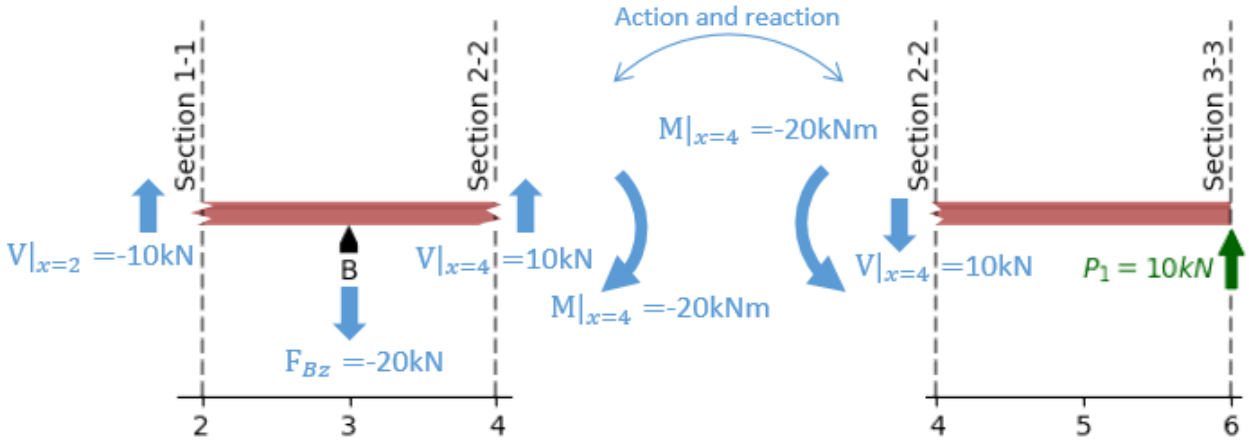


Fig. 7: Subsystem between planes 1-1 (btw supports *A* and *B*) and 3-3 (beam's right end)

As an exercise, you can follow this procedure and calculate the general result for an arbitrary x-coordinate. You should obtain a function like the one shown in the image below.

Show/Hide code for generating the image below

```
my_beam = Beam(6)
my_beam.rolling_support = 0  # x-coordinate of the rolling support
my_beam.pinned_support = 3    # x-coordinate of the pinned support
my_beam.add_loads([PointLoadV(10, 6)])  # 10 kN pointing upwards, at x=6 m
fig = plt.figure()
my_beam.plot_shear_force(fig.add_subplot(2, 1, 1))
my_beam.plot_bending_moment(fig.add_subplot(2, 1, 2))
```

../../examples/example_shear1.png

CHAPTER 4

Relationship between shear force and bending moment

The analysis in the section above was restricted to point loads in order to keep it simple. However, it applies universally for distributed loads as well, as we are going to see next.

It may not be obvious at first sight, but the functions corresponding to shear force $\mathbf{V}(\mathbf{x})$ and bending moment $\mathbf{M}(\mathbf{x})$ are intimately correlated (i.e. you can use one of them for calculating the other one). We are going to prove this by performing the same analysis explained above to a differential beam segment of length Δx.

By this point, our analysis methodology should be clear. It consists of the following steps:

1. Choosing an arbitrary sector of a beam.
2. Drawing a free-body diagram of the partition.
3. Applying Newton's first law to the free-body diagram.

Let's consider an arbitrarily loaded beam as shown in the figure below, where $\mathbf{P_z}(x)$ is an arbitrarily distributed load applied to the beam.

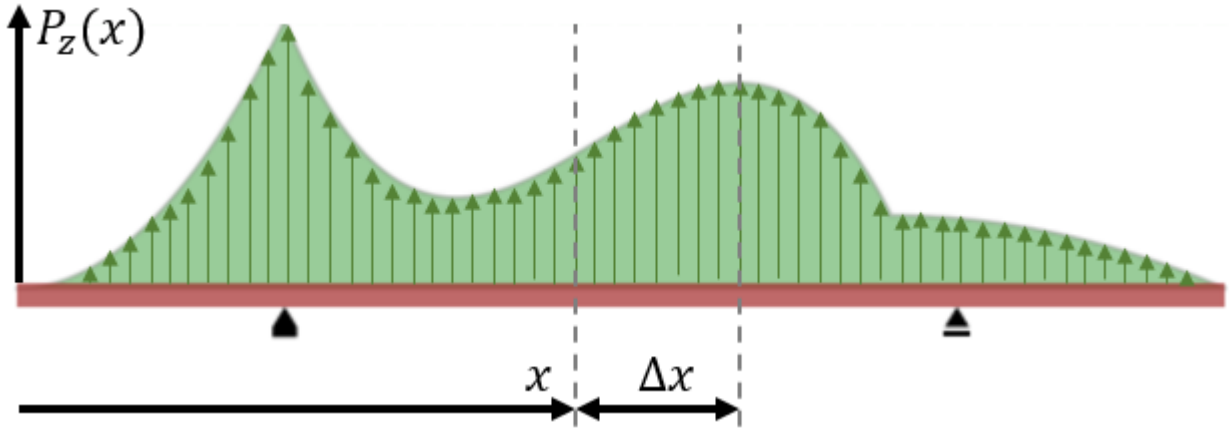


Fig. 1: Generic problem consisting of a beam under an arbitrarily distributed load $\mathbf{P_z}(x)$

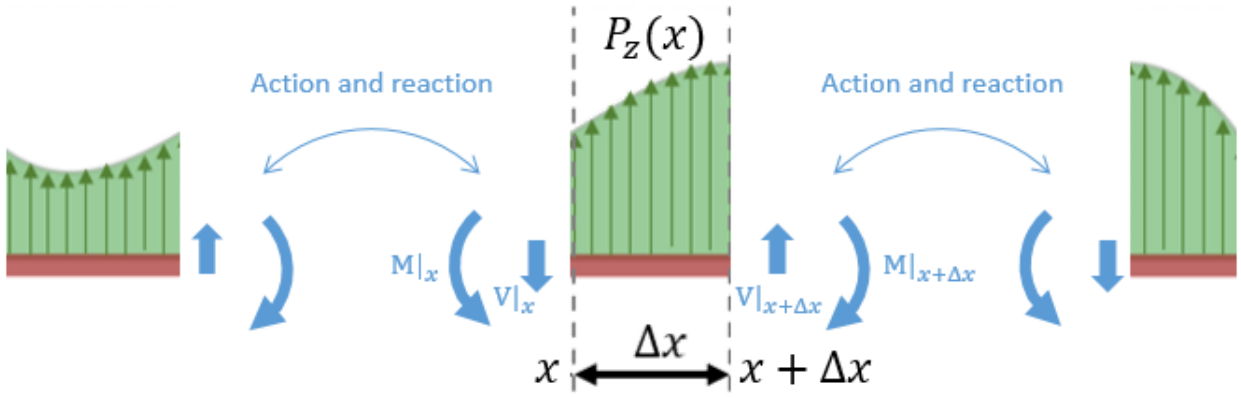


Fig. 2: Free body diagram of a differential segment of the beam

Let's zoom-in and draw a free-body diagram of a given beam segment $[x_0 \leq x \leq x_0 + \Delta x]$.

The equilibrium of vertical forces yields then the following:

$$\begin{aligned} \sum \mathbf{F_z} &= 0 \\ V(x+\Delta x) - V(x) + \overline{\mathbf{P_z}}(x)\Delta x &= 0 \\ \frac{V(x+\Delta x) - V(x)}{\Delta x} &= -\overline{\mathbf{P_z}}(x) \\ \lim_{\Delta x \to 0} \qquad \boxed{\frac{dV(x)}{dx} = -\mathbf{P_z}(x)} & \end{aligned}$$

which in words means that the rate of change of the shear force is equal to (minus) the value of the distributed load acting on a given x-coordinate.

Note that in the expression above, $\overline{\mathbf{P_z}}(x)$ represents the average value of $\mathbf{P_z}(x)$ in the given interval. As Δx goes to zero, $\overline{\mathbf{P_z}}(x)$ converges to $\mathbf{P_z}(x)$.

The equilibrium of moments can be written as:

$$\begin{aligned} \sum \mathbf{M}|_{\mathbf{x}+\mathbf{\Delta x}} &= 0 \\ M(x+\Delta x) - M(x) - V(x)\Delta x + (\overline{\mathbf{P_z}}(x)\Delta x \cdot \tfrac{\Delta x}{2}) &= 0 \\ \frac{M(x+\Delta x) - M(x)}{\Delta x} + \underbrace{(\overline{\mathbf{P_z}}(x)\Delta x \cdot \frac{\Delta x}{2})}_{\Delta x^2 \to 0} &= V(x) \\ \lim_{\Delta x \to 0} \qquad \boxed{\frac{dM(x)}{dx} = V(x)} & \end{aligned}$$

which presents explicitly the relationship between $M(x)$ and $V(x)$: the rate of change of the bending moment at a given point is equal to the shear force at that point.

Ok, that was it. For a worked example, take a look at the next section: *Beam with two point loads and two distributed loads*.

CHAPTER 5

Beam with two point loads and two distributed loads

This example demonstrates some functionality of the beambending package.

Try it in online:

5.1 Specifications

A beam resting on two supports at x=2 m and x=7 m, with the following applied loads:

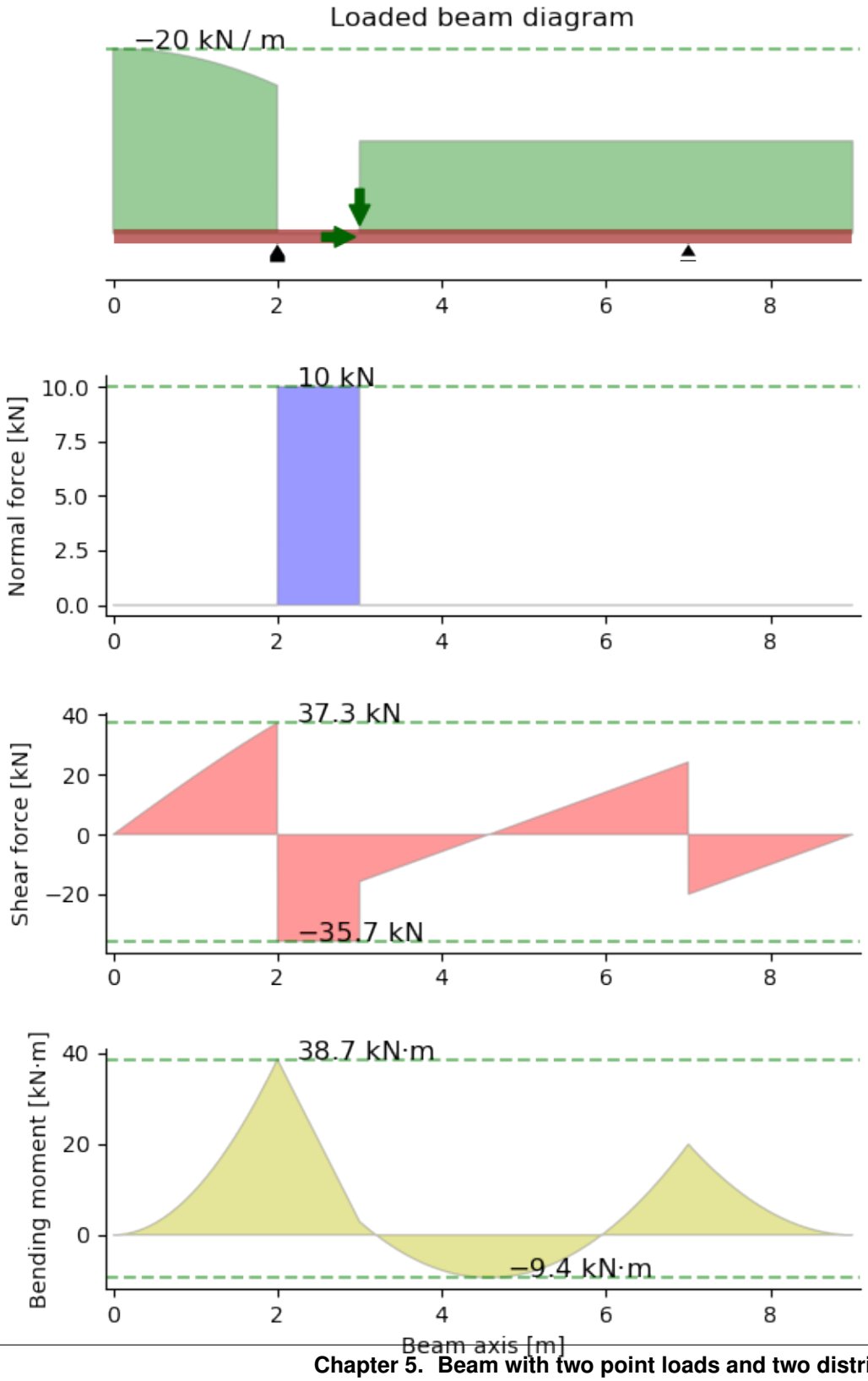
- a downward force of 20 kN at x=3 m;
- a force of 10 kN pointing right, also at x=3 m;
- a downward distributed load of 20 kN/m on 0 <= x <= 2 m;
- a downward distributed load of 10 kN/m on 3 m <= x <= 9 m

5.2 Results

The figure below shows the load case corresponding to the description above. The resulting diagrams are respectively:

1. a schematic of the beam and its applied (external) loads[1] ,
2. a normal force diagram: $\mathbf{N}(\mathbf{x})$,
3. a shear force diagram $\mathbf{V}(\mathbf{x})$, and
4. a bending moment diagram $\mathbf{M}(\mathbf{x})$.

[1] The x-coordinates of the point loads are marked with green arrows, but magnitudes are not displayed in the beam schematic in order to keep the figure clean and simple.


Loaded beam diagram
−20 kN / m
0
2
4
6
8
10 kN
Normal force [kN]
10.0
7.5
5.0
2.5
0.0
37.3 kN
Shear force [kN]
40
20
0
−20
−35.7 kN
38.7 kN·m
Bending moment [kN·m]
−9.4 kN·m
Beam axis [m]

5.3 Code

```
# -*- coding: utf-8 -*-
"""Example 1: Shear force and bending moment diagrams for a beam subjected to
both point loads and distributed loads.
"""
import os
from beambending import Beam, DistributedLoadV, PointLoadH, PointLoadV, x

def example_1():
    """Run example 1"""
    beam = Beam(9)  # Initialize a Beam object of length 9 m
    beam.pinned_support = 2    # x-coordinate of the pinned support
    beam.rolling_support = 7  # x-coordinate of the rolling support
    beam.add_loads((
                    PointLoadH(10, 3),  # 10 kN pointing right, at x=3 m
                    PointLoadV(-20, 3),  # 20 kN downwards, at x=3 m
                    DistributedLoadV(-10, (3, 9)),  # 10 kN/m, downwards, for 3 m <=
↪x <= 9 m
                    DistributedLoadV(-20 + x**2, (0, 2)),  # variable load, for 0 <=
↪x <= 2 m
                ))
    fig = beam.plot()

    # save the png and add it to the documentation
    mod_path = os.path.dirname(os.path.abspath(__file__))  # current module
    save_name = os.path.basename(__file__).replace('.py', '.png')  # file name
    save_path = os.path.join(mod_path, save_name)
    fig.savefig(save_path, transparent=True)


if __name__ == '__main__':  # call function when run as script
    example_1()
```

CHAPTER 6

Beambending Reference

Main module containing the main class Beam, and auxiliary classes PointLoadH, PointLoadV, DistributedLoadH, and DistributedLoadV.

6.1 Example

```
>>> my_beam = Beam(9)
>>> my_beam.pinned_support = 2     # x-coordinate of the pinned support
>>> my_beam.rolling_support = 7  # x-coordinate of the rolling support
>>> my_beam.add_loads([PointLoadV(-20, 3)])  # 20 kN downwards, at x=3 m
>>> print("(F_Ax, F_Ay, F_By) =", my_beam.get_reaction_forces())
(F_Ax, F_Ay, F_By) = (0.0, 16.0, 4.0)
```

6.2 Beam

class beam.**Beam**(*span: float = 10*)

Represents a one-dimensional beam that can take axial and tangential loads.

Through the method *add_loads*, a Beam object can accept a list of:

- PointLoad objects, and/or
- DistributedLoad objects.

Notes

- The Beam class currently supports only statically determined beams with (exactly) one pinned and one roller support.
- The default units package units for length, force and bending moment (torque) are respectively (m, kN, kN·m)

beam.Beam.**add_loads**(*self*, *loads: list*)
Apply an arbitrary list of (point- or distributed) loads to the beam.

Parameters

loads [iterable] An iterable containing DistributedLoad or PointLoad objects to be applied to the Beam object. Note that the load application point (or segment) must be within the Beam span.

beam.Beam.**get_reaction_forces**(*self*)
Calculates the reaction forces at the supports, given the applied loads.

The first and second values correspond to the horizontal and vertical forces of the pinned support. The third one is the vertical force at the rolling support.

Returns

F_Ax, F_Ay, F_By: (float, float, float) reaction force components for pinned (x,y) and rolling (y) supports respectively.

beam.Beam.**plot**(*self*)
Generates a single figure with 4 plots corresponding respectively to:

- a schematic of the loaded beam
- normal force diagram,
- shear force diagram, and
- bending moment diagram.

These plots can be generated separately with dedicated functions.

Returns

figure [*~matplotlib.figure.Figure*] Returns a handle to a figure with the 3 subplots: Beam schematic, shear force diagram, and bending moment diagram.

beam.Beam.**plot_beam_diagram**(*self*, *ax=None*)
Returns a schematic of the beam and all the loads applied on it.

beam.Beam.**plot_normal_force**(*self*, *ax=None*)
Returns a plot of the normal force as a function of the x-coordinate.

beam.Beam.**plot_shear_force**(*self*, *ax=None*)
Returns a plot of the shear force as a function of the x-coordinate.

beam.Beam.**plot_bending_moment**(*self*, *ax=None*)
Returns a plot of the bending moment as a function of the x-coordinate.

6.3 PointTorque

class beam.**PointTorque**
Point clockwise torque, described by a tuple of floats: (torque, coord).

Examples

```
>>> motor_torque = PointTorque(30, 4)  # 30 kN·m (clockwise) torque at x=4 m
```

6.4 PointLoad

class beam.**PointLoadH**
Horizontal point load described by a tuple of floats: (force, coord).

Examples

```
>>> external_force = PointLoadH(10, 9)  # 10 kN towards the right at x=9 m
>>> external_force
PointLoadH(force=10, coord=9)
```

class beam.**PointLoadV**
Vertical point load described by a tuple of floats: (force, coord).

Examples

```
>>> external_force = PointLoadV(-30, 3)  # 30 kN downwards at x=3 m
>>> external_force
PointLoadV(force=-30, coord=3)
```

6.5 DistributedLoad

class beam.**DistributedLoadH**
Distributed horizontal load, described by its functional form and application interval.

Examples

```
>>> wind_load = DistributedLoadH("10*x**2+5", (1, 3))  # Quadratically growing␣
↪load for 1<y<3
```

class beam.**DistributedLoadV**
Distributed vertical load, described by its functional form and application interval.

Examples

```
>>> snow_load = DistributedLoadV("10*x+5", (0, 2))  # Linearly growing load for 0
↪<x<2 m
```

Python Module Index

b

Index