FEMethods Documentation

Release 0.1.7a1

Joseph Contreras

Oct 17, 2020

Table of Contents:

FEmethods is a python package designed to analyse structural elements using Finite Element Methods (FEM) to solve for element reaction forces and calculate the deflection of the element.

Using FEM has the advantage over closed form (exact) equations because it uses numerical techniques that can easily be used on many different load cases, including statically indeterminate cases. The disadvantage of FEM is that it will have less accuracy then the exact equations derived for a particular case.

Note: This package is currently a work-in-progress.

CHAPTER 1

Introduction

This is the introduction to FEmethods. It will introduce Finite Element Methods in general, and then give a few examples of how to use FEmethods.

The code can be found on github.com

1.1 Installation

FEmethods is hosted on PyPi, so installation is straightforward.

```
>>>pip install femethods
```

It can also be installed from source.

```
>>>git clone https://github.com/josephjcontreras/FEmethods.git
```

Then to test that the installation worked properly, you can try this simple example case of a simply supported beam with a single, centered point load.

```
>>>from femethods.elements import Beam
>>>from femethods.reactions import PinnedReaction
>>>from femethods.loads import PointLoad

>>>b = Beam(30, loads=[PointLoad(-100, 15)], reactions=[PinnedReaction(x)␣
↪for x in [0, 30]])

>>>b.solve()
>>>print(b)

PARAMETERS
Length (length): 30
Young's Modulus (E): 1
Area moment of inertia (Ixx): 1
```

(continues on next page)

(continued from previous page)

```
LOADING
Type: point load
    Location: 15
   Magnitude: -100
REACTIONS
Type: pinned
    Location: 0
       Force: 50.0
      Moment: 0.0
Type: pinned
    Location: 30
       Force: 50.0
      Moment: 0.0
```

CHAPTER 2

FEmethods Package

This section will go into details on how each module should be called.

2.1 femethods.elements module

The elements module contains finite element classes

Currently the only element that is defined is a beam element.

class `femethods.elements.`**`Beam`**(*length: float, loads: List[Load], reactions: List[Reaction], E: float = 1, Ixx: float = 1*)

Bases: `femethods.core._base_elements.BeamElement`

A Beam defines a beam element for analysis

A beam element is a slender member that is subjected to transverse loading. It is assumed to have homogeneous properties, with a constant cross-section.

Parameters

- **length** (`float`) – the length of a beam. This is the total length of the beam, this is not the length of the meshed element. This must be a float that is greater than 0.
- **loads** (`list`) – list of load elements
- **reactions** (`list`) – list of reactions acting on the beam
- **E** (`float`, optional) – Young's modulus of the beam in units of $\frac{force}{length^2}$. Defaults to 1. The $force$ units used here are the same units that are used in the input forces, and calculated reaction forces. The $length$ unit must be the same as the area moment of inertia (**Ixx**) and the beam **length** units.
- **Ixx** (`float`, optional) – Area moment of inertia of the beam. Defaults to 1. This is constant (constant cross-sectional area) along the length of the beam. This is in units of $length^4$. This must be the same length unit of Young's modulus (**E**) and the beam **length**.

bending_stress (*x*, *dx=1*, *c=1*)
returns the bending stress at global coordinate x

Deprecated since version 0.1.7a1: calculate bending stress as `Beam.moment(x) * c / Ixx`

deflection (*x: float*) → numpy.float64
Calculate deflection of the beam at location x

Parameters **x** (`float | int`) – location along the length of the beam where deflection should be calculated.

Returns deflection of the beam in units of the beam length

Return type `float`

Raises

- `ValueError` – when the $0 \leq x \leq length$ is False
- `TypeError` – when x cannot be converted to a float

moment (*x: float*, *dx: float = 1e-05*, *order: int = 9*) → numpy.float64
Calculate the moment at location x

Calculate the moment in the beam at the global x value by taking the second derivative of the deflection curve.

$$M(x) = E \cdot Ixx \cdot \frac{d^2 v(x)}{dx^2}$$

where M is the moment, E is Young's modulus and Ixx is the area moment of inertia.

Parameters

- **x** (`int`) – location along the beam where moment is calculated
- **dx** (`float`, optional) – spacing. Default is 1e-5
- **order** (`int`, optional) – number of points to use, must be odd. Default is 9

Returns moment in beam at location x

Return type `float`

Raises

- `ValueError` – when the $0 \leq x \leq length$ is False
- `TypeError` – when x cannot be converted to a float

For more information on the parameters, see the scipy.misc.derivative documentation.

plot (*n=250*, *title='Beam Analysis'*, *diagrams=None*, *diagram_labels=None*, ***kwargs*)
Plot the deflection, moment, and shear along the length of the beam

The plot method will create a `matplotlib.pyplot` figure with the deflection, moment, and shear diagrams along the length of the beam element. Which of these diagrams, and their order may be customized.

Parameters

- **n** (`int`) – defaults to *250*: number of data-points to use in plots
- **title** (`str`) – title on top of plot
- **diagrams** (`tuple`) – defaults to *('shear', 'moment', 'deflection')* tuple of diagrams to plot. All values in tuple must be strings, and one of the defaults. Valid values are `('shear', 'moment', 'deflection')`

- **diagram_labels** (`tuple`) – y-axis labels for subplots. Must have the same length as *diagrams*

Returns Tuple of `matplotlib.pyplot` figure and list of axes in the form `(figure, axes)`

Return type `tuple`

Note: The plot method will create the figure handle, but will not automatically show the figure. To show the figure use *`Beam.show()`* or `matplotlib.pyplot.show()`

Changed in version 0.1.7a1: Removed *`bending_stress`* parameter

Changed in version 0.1.7a1: Added `diagrams` and `diagram_labels` parameters

`shear`(*x: float*, *dx: float = 0.01*, *order: int = 5*) → numpy.float64
Calculate the shear force in the beam at location x

Calculate the shear in the beam at the global x value by taking the third derivative of the deflection curve.

$$V(x) = E \cdot Ixx \cdot \frac{d^3 v(x)}{dx^3}$$

where V is the shear force, E is Young's modulus and Ixx is the area moment of inertia.

Parameters

- **x** (`int`) – location along the beam where moment is calculated
- **dx** (`float`, optional) – spacing. Default is 0.01
- **order** (`int`, optional) – number of points to use, must be odd. Default is 5

Returns moment in beam at location x

Return type `float`

Raises

- `ValueError` – when the $0 \leq x \leq length$ is False
- `TypeError` – when x cannot be converted to a float

For more information on the parameters, see the scipy.misc.derivative documentation.

`static show`(**args*, ***kwargs*) → None
Wrapper function for showing matplotlib figure

This method gives direct access to the matplotlib.pyplot.show function so the calling code is not required to import matplotlib directly just to show the plots

Parameters **args/kwargs** – args and kwargs are passed directly to matplotlib.pyplot.show

2.2 femethods.loads module

Module to define different loads

class `femethods.loads.`**`Load`**(*magnitude: Optional[float]*, *location: float = 0*)
Bases: `femethods.core._common.Forces`

Base class for all load types

Used primarily for type checking the loads on input

name = ''

class femethods.loads.**MomentLoad**(*magnitude: float*, *location: float*)

Bases: *femethods.loads.Load*

class specific to a moment load

name = 'moment load'

class femethods.loads.**PointLoad**(*magnitude: Optional[float], location: float*)

Bases: *femethods.loads.Load*

class specific to a point load

name = 'point load'

2.3 femethods.mesh module

Mesh module that will define the mesh.

class femethods.mesh.**Mesh**(*length: float, loads: List[Load], reactions: List[Reaction], dof: int*)

Bases: object

define a mesh that will handle degrees-of-freedom (dof), element lengths etc.

the input degree-of-freedom (dof) parameter is the degrees-of-freedom for a single element

dof

Degrees of freedom of the entire beam

Returns Read-only. Number of degrees of freedom of the beam

Return type int

lengths

List of lengths of mesh elements

Returns Read-only. List of lengths of local mesh elements

Return type list

nodes

num_elements

Number of mesh elements

Returns Read-only. Number of elements in mesh

Return type int

2.4 femethods.reactions module

The reactions module defines different reaction classes

A reaction is required to support an element to resist any input forces.

There are two types of reactions that are defined.

- PinnedReaction, allows rotational displacement only
- FixedReaction, does not allow any displacement

class `femethods.reactions.`**`FixedReaction`**(*location: float*)

Bases: *`femethods.reactions.Reaction`*

A FixedReaction does not allow any displacement or change in angle

A FixedReaction resists both force and moments. The displacement and the angle are both constrained and must be zero at the reaction point. FixedReactions are typically applied at the ends of a Beam.

Parameters **`location`** (`float`) – the axial location of the reaction along the length of the beam

`name`

short name of the reaction (fixed). Used internally

Type `str`

> **Warning:** The **name** attribute is used internally. **Do not change this value!**

`name = 'fixed'`

class `femethods.reactions.`**`PinnedReaction`**(*location: float*)

Bases: *`femethods.reactions.Reaction`*

A PinnedReaction allows rotation displacements only

A PinnedReaction represents a pinned, frictionless pivot that can resist motion both normal and axial directions to the beam. It will not resist moments. The deflection of a beam at the PinnedReaction is always zero, but the angle is free to change

Parameters **`location`** (`float`) – the axial location of the reaction along the length of the beam

`name`

short name of the reaction (pinned). Used internally

Type `str`

> **Warning:** The **name** attribute is used internally. **Do not change this value!**

`name = 'pinned'`

class `femethods.reactions.`**`Reaction`**(*location: float*)

Bases: `femethods.core._common.Forces`

Base class for all reactions

The Reaction class defines general properties related to all reaction types.

Parameters **`location`** (`float`) – the axial location of the reaction along the length of the beam.

> **Note:** Any force or moment values that where calculated values are invalidated (set to `None`) any time the location is set.

`force`

the force of the reaction after it has been calculated

Type `float | None`

`moment`

The moment of the reaction after it has been calculated

Type `float | None`

`boundary`

`invalidate()` → None

Invalidate the reaction values

This will set the force and moment values to `None`

To be used whenever the parameters change and the reaction values are no longer valid.

`location`

Location of the reaction along the length of the beam

The units of the length property is the same as the units of the beam length.

The value of the location must be a positive value that is less than or equal to the length of the beam, or it will raise a ValueError.

Note: The force and moment values are set to `None` any time the location is set.

`name = ''`

`value`

Simple tuple of force and moment

Returns `tuple` (force, moment)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index