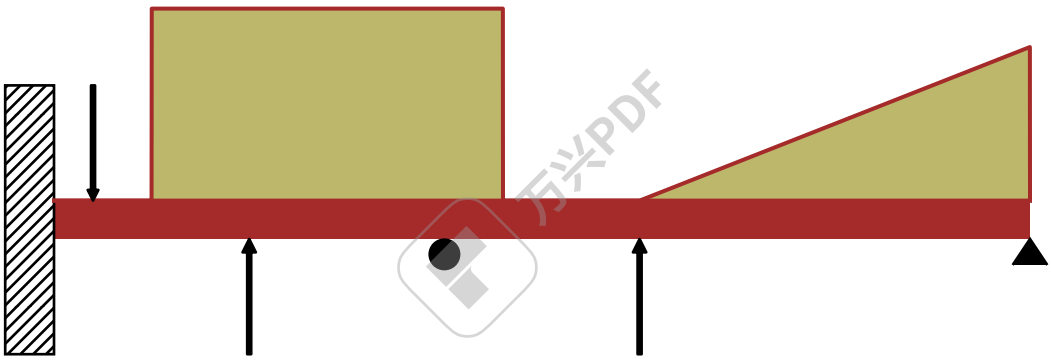


Examples

There is a cantilever beam of length 4 meters. For first 2 meters its moment of inertia is $1.5*I$ and I for the other end. A pointload of magnitude 4 N is applied from the top at its free end.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
>>> R1, R2 = symbols('R1, R2')
>>> b1 = Beam(2, E, 1.5*I)
>>> b2 = Beam(2, E, I)
>>> b = b1.join(b2, "fixed")
>>> b.apply_load(20, 4, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 0, -2)
>>> b.bc_slope = [(0, 0)]
>>> b.bc_deflection = [(0, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.load
80*SingularityFunction(x, 0, -2) - 20*SingularityFunction(x, 0, -1) +␣
↪20*SingularityFunction(x, 4, -1)
>>> b.slope()
(-((-80*SingularityFunction(x, 0, 1) + 10*SingularityFunction(x, 0,␣
↪2) - 10*SingularityFunction(x, 4, 2))/I + 120/I)/E + 80.0/
↪(E*I))*SingularityFunction(x, 2, 0)
- 0.666666666666667*(-80*SingularityFunction(x, 0, 1) +␣
↪10*SingularityFunction(x, 0, 2) - 10*SingularityFunction(x, 4,␣
↪2))*SingularityFunction(x, 0, 0)/(E*I)
+ 0.666666666666667*(-80*SingularityFunction(x, 0, 1) +␣
↪10*SingularityFunction(x, 0, 2) - 10*SingularityFunction(x, 4,␣
↪2))*SingularityFunction(x, 2, 0)/(E*I)
```

property length

Length of the Beam.

property load

Returns a Singularity Function expression which represents the load distribution curve of the Beam object.

Examples

There is a beam of length 4 meters. A moment of magnitude 3 Nm is applied in the clockwise direction at the starting point of the beam. A point load of magnitude 4 N is applied from the top of the beam at 2 meters from the starting point and a parabolic ramp load of magnitude 2 N/m is applied below the beam starting from 3 meters away from the starting point of the beam.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
>>> b = Beam(4, E, I)
```

(continues on next page)

(continued from previous page)

```
>>> b.apply_load(-3, 0, -2)
>>> b.apply_load(4, 2, -1)
>>> b.apply_load(-2, 3, 2)
>>> b.load
-3*SingularityFunction(x, 0, -2) + 4*SingularityFunction(x, 2, -1) -␣
↪2*SingularityFunction(x, 3, 2)
```

max_bmoment()

Returns maximum Shear force and its coordinate in the Beam object.

max_deflection()

Returns point of max deflection and its corresponding deflection value in a Beam object.

max_shear_force()

Returns maximum Shear force and its coordinate in the Beam object.

plot_bending_moment(*subs=None*)

Returns a plot for Bending moment present in the Beam object.

Parameters

subs : dictionary

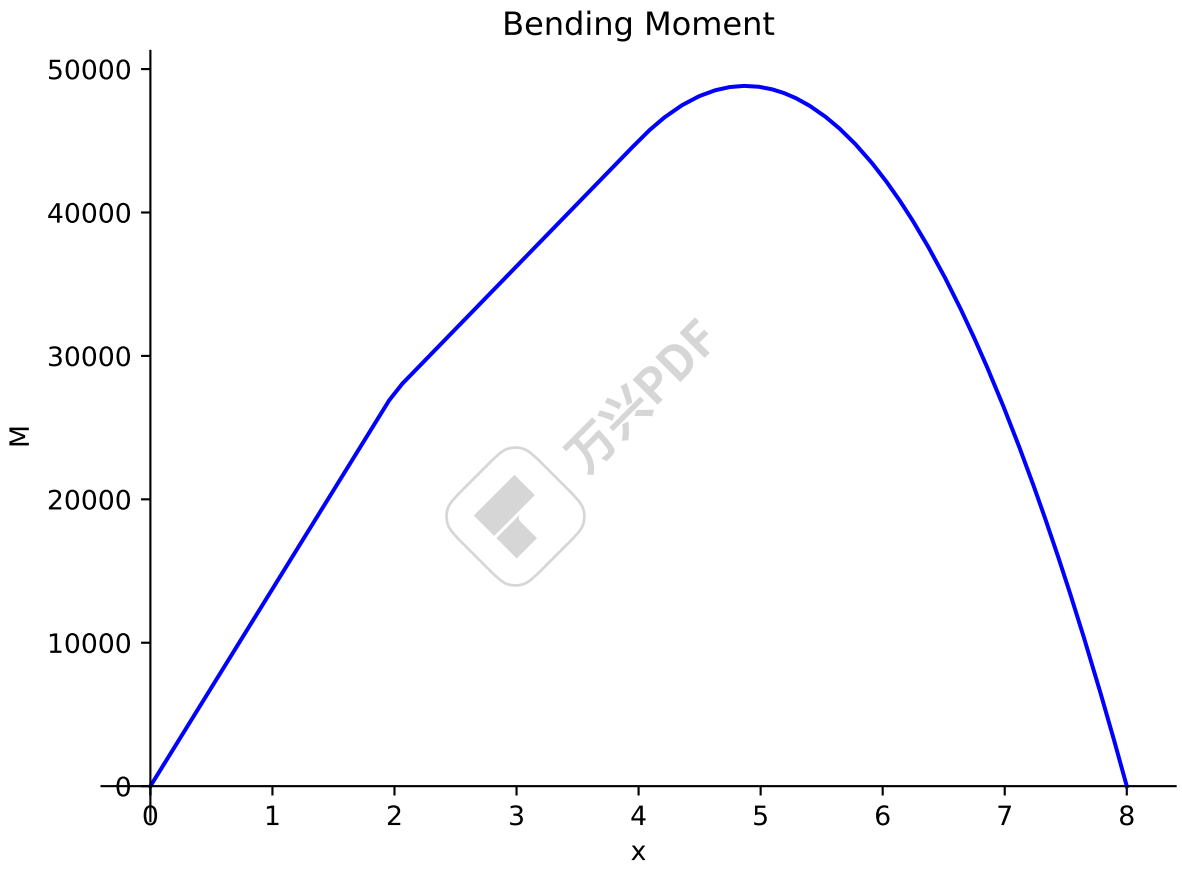
Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 8 meters. A constant distributed load of 10 KN/m is applied from half of the beam till the end. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. A pointload of magnitude 5 KN is also applied from top of the beam, at a distance of 4 meters from the starting point. Take E = 200 GPa and I = 400*(10**-6) meter**4.

Using the sign convention of downwards forces being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(8, 200*(10**9), 400*(10**-6))
>>> b.apply_load(5000, 2, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 8, -1)
>>> b.apply_load(10000, 4, 0, end=8)
>>> b.bc_deflection = [(0, 0), (8, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.plot_bending_moment()
Plot object containing:
[0]: cartesian line: 13750*SingularityFunction(x, 0, 1) -␣
↪5000*SingularityFunction(x, 2, 1)
- 5000*SingularityFunction(x, 4, 2) + 31250*SingularityFunction(x, 8,␣
↪1)
+ 5000*SingularityFunction(x, 8, 2) for x over (0.0, 8.0)
```


Bending Moment
50000
40000
30000
20000
10000
0
M
0
1
2
3
4
5
6
7
8
x

plot_deflection(*subs=None*)

Returns a plot for deflection curve of the Beam object.

Parameters

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 8 meters. A constant distributed load of 10 KN/m is applied from half of the beam till the end. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. A pointload of magnitude 5 KN is also applied from top of the beam, at a distance of 4 meters from the starting point. Take E = 200 GPa and I = 400*(10**-6) meter**4.

Using the sign convention of downwards forces being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(8, 200*(10**9), 400*(10**-6))
>>> b.apply_load(5000, 2, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 8, -1)
>>> b.apply_load(10000, 4, 0, end=8)
>>> b.bc_deflection = [(0, 0), (8, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.plot_deflection()
Plot object containing:
[0]: cartesian line: 0.0013854166666666667*x - 2.86458333333333e-
 ↪5*SingularityFunction(x, 0, 3)
+ 1.04166666666667e-5*SingularityFunction(x, 2, 3) + 5.
 ↪20833333333333e-6*SingularityFunction(x, 4, 4)
- 6.51041666666667e-5*SingularityFunction(x, 8, 3) - 5.
 ↪20833333333333e-6*SingularityFunction(x, 8, 4)
for x over (0.0, 8.0)
```

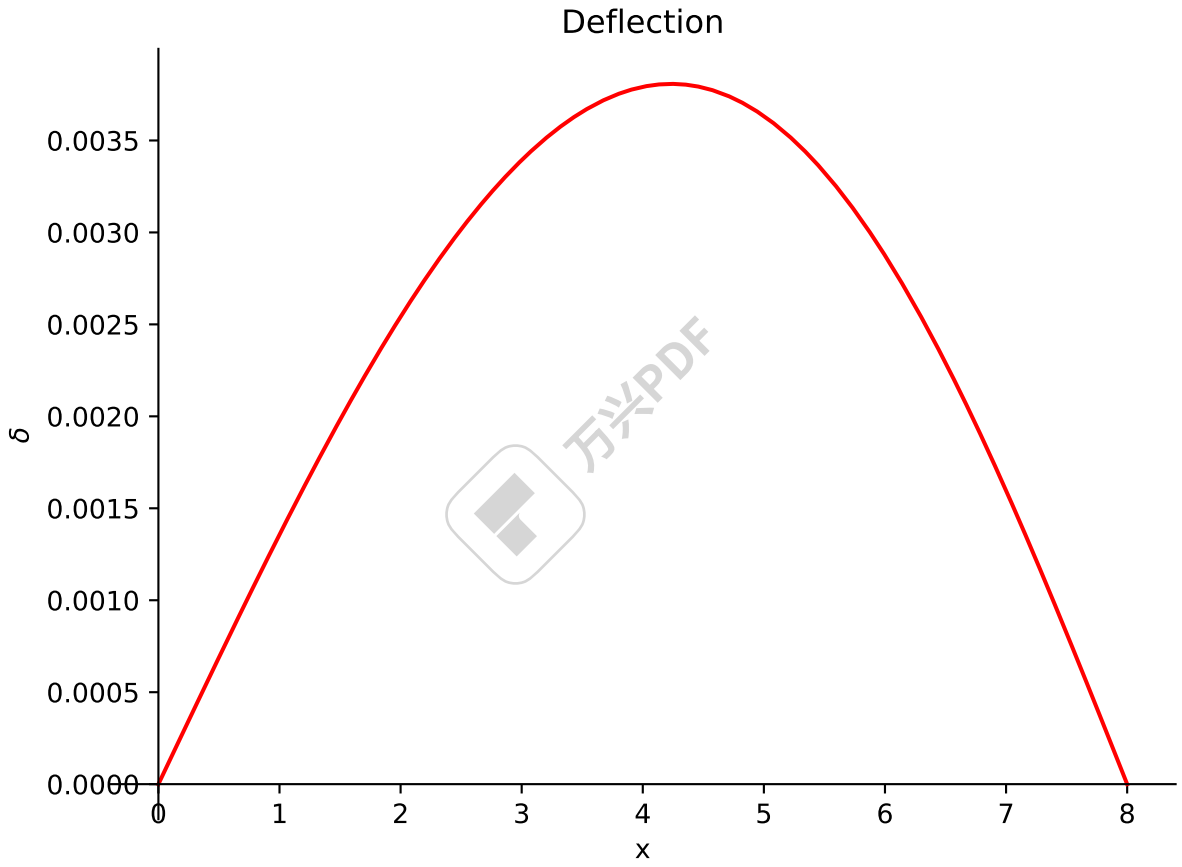
plot_ild_moment(*subs=None*)

Plots the Influence Line Diagram for Moment under the effect of a moving load. This function should be called after calling solve_for_ild_moment().

Parameters

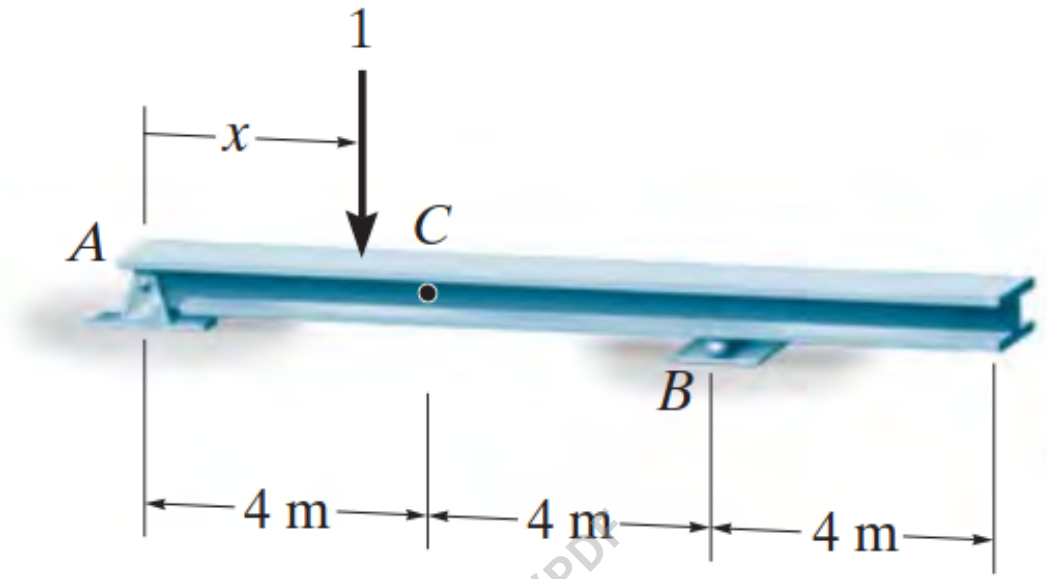
subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.


Deflection
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
δ
0
1
2
3
4
5
6
7
8
x

Examples

There is a beam of length 12 meters. There are two simple supports below the beam, one at the starting point and another at a distance of 8 meters. Plot the I.L.D. for Moment at a distance of 4 meters under the effect of a moving load of magnitude 1kN.



Using the sign convention of downwards forces being positive.

```
>>> from sympy import symbols
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> E, I = symbols('E, I')
>>> R_0, R_8 = symbols('R_0, R_8')
>>> b = Beam(12, E, I)
>>> b.apply_support(0, 'roller')
>>> b.apply_support(8, 'roller')
>>> b.solve_for_ild_reactions(1, R_0, R_8)
>>> b.solve_for_ild_moment(4, 1, R_0, R_8)
>>> b.ild_moment
Piecewise((-x/2, x < 4), (x/2 - 4, x > 4))
>>> b.plot_ild_moment()
Plot object containing:
[0]: cartesian line: Piecewise((-x/2, x < 4), (x/2 - 4, x > 4)) for x␣
↪over (0.0, 12.0)
```

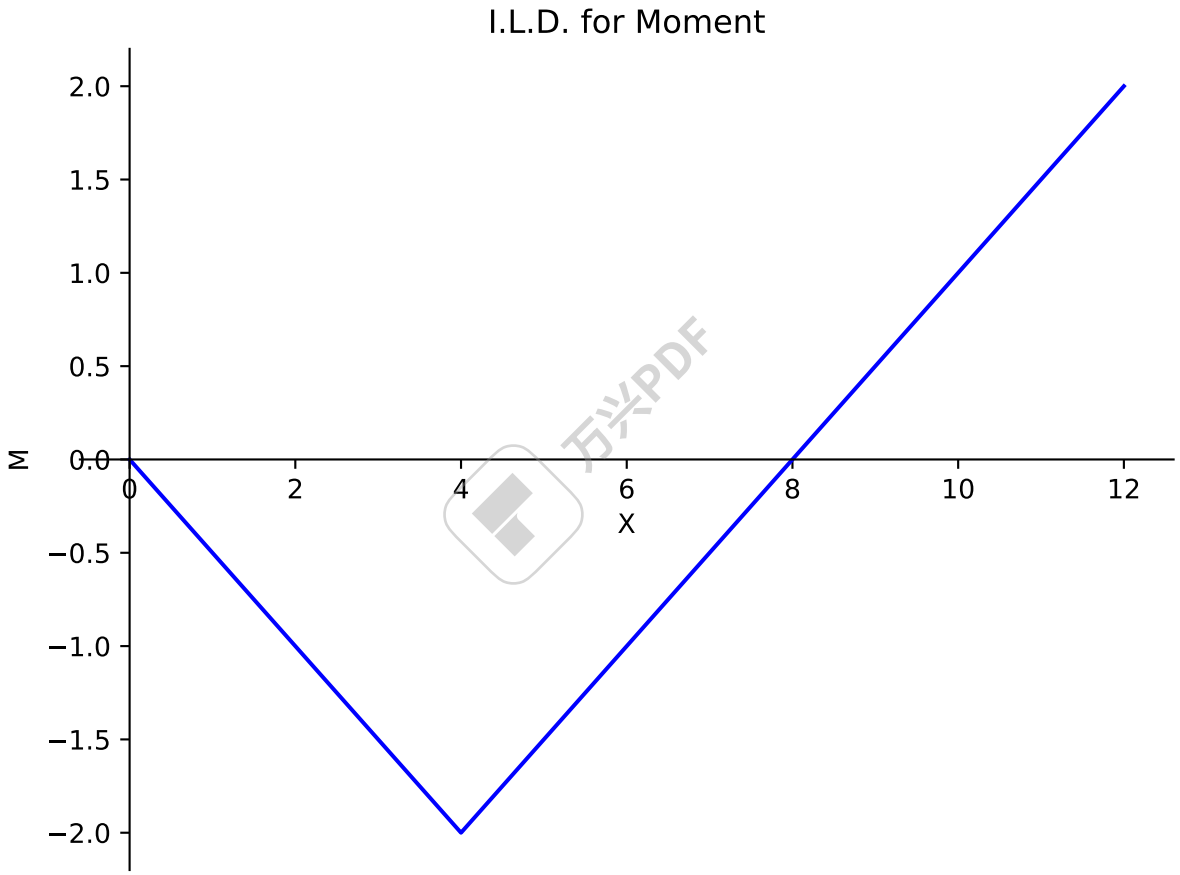
plot_ild_reactions(*subs=None*)

Plots the Influence Line Diagram of Reaction Forces under the effect of a moving load. This function should be called after calling solve_for_ild_reactions().

Parameters

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.


I.L.D. for Moment
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
M
0
2
4
6
8
10
12
X

Examples

There is a beam of length 10 meters. A point load of magnitude 5KN is also applied from top of the beam, at a distance of 4 meters from the starting point. There are two simple supports below the beam, located at the starting point and at a distance of 7 meters from the starting point. Plot the I.L.D. equations for reactions at both support points under the effect of a moving load of magnitude 1kN.

Using the sign convention of downwards forces being positive.

```
>>> from sympy import symbols
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> E, I = symbols('E, I')
>>> R_0, R_7 = symbols('R_0, R_7')
>>> b = Beam(10, E, I)
>>> b.apply_support(0, 'roller')
>>> b.apply_support(7, 'roller')
>>> b.apply_load(5,4,-1)
>>> b.solve_for_ild_reactions(1,R_0,R_7)
>>> b.ild_reactions
{R_0: x/7 - 22/7, R_7: -x/7 - 20/7}
>>> b.plot_ild_reactions()
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: x/7 - 22/7 for x over (0.0, 10.0)
Plot[1]:Plot object containing:
[0]: cartesian line: -x/7 - 20/7 for x over (0.0, 10.0)
```

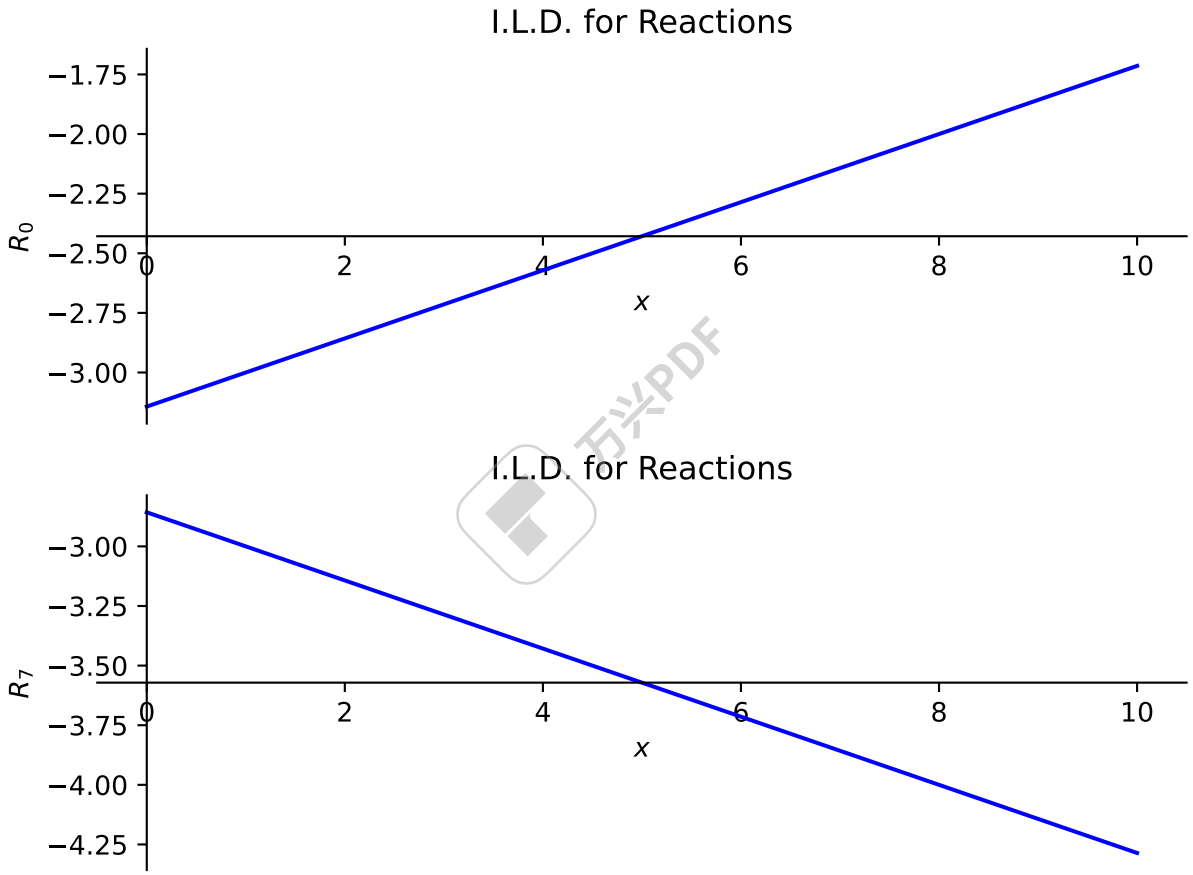
plot_ild_shear(*subs=None*)

Plots the Influence Line Diagram for Shear under the effect of a moving load. This function should be called after calling solve_for_ild_shear().

Parameters

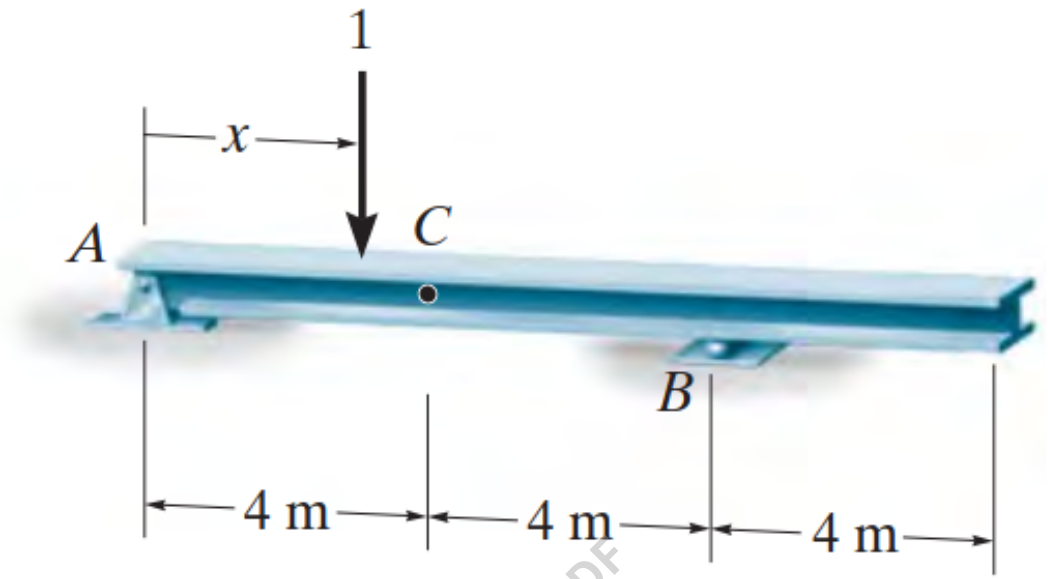
subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.


I.L.D. for Reactions
−1.75
−2.00
−2.25
−2.50
−2.75
−3.00
R_0
0
2
4
6
8
10
x
I.L.D. for Reactions
−3.00
−3.25
−3.50
−3.75
−4.00
−4.25
R_7
0
2
4
6
8
10
x

Examples

There is a beam of length 12 meters. There are two simple supports below the beam, one at the starting point and another at a distance of 8 meters. Plot the I.L.D. for Shear at a distance of 4 meters under the effect of a moving load of magnitude 1kN.



Using the sign convention of downwards forces being positive.

```
>>> from sympy import symbols
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> E, I = symbols('E, I')
>>> R_0, R_8 = symbols('R_0, R_8')
>>> b = Beam(12, E, I)
>>> b.apply_support(0, 'roller')
>>> b.apply_support(8, 'roller')
>>> b.solve_for_ild_reactions(1, R_0, R_8)
>>> b.solve_for_ild_shear(4, 1, R_0, R_8)
>>> b.ild_shear
Piecewise((x/8, x < 4), (x/8 - 1, x > 4))
>>> b.plot_ild_shear()
Plot object containing:
[0]: cartesian line: Piecewise((x/8, x < 4), (x/8 - 1, x > 4)) for x␣
↪over (0.0, 12.0)
```

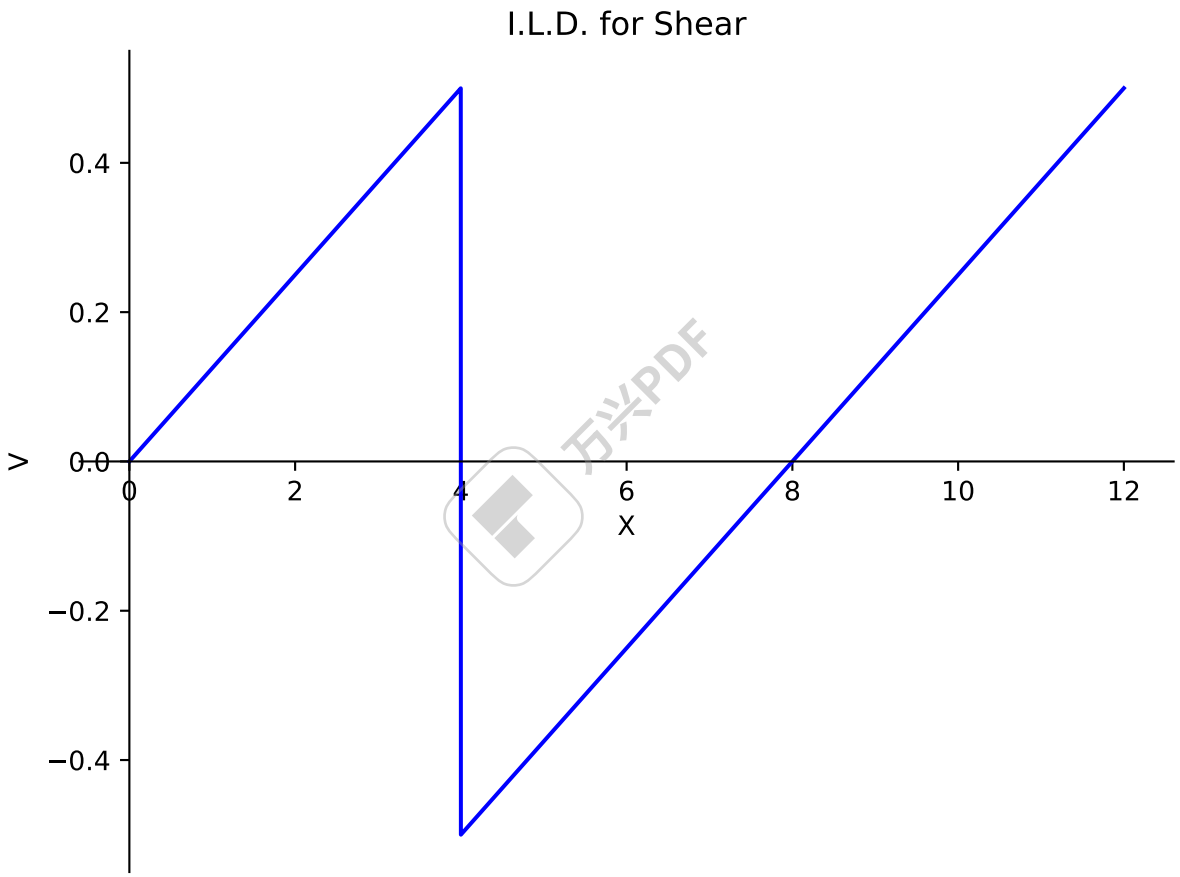
plot_loading_results(*subs=None*)

Returns a subplot of Shear Force, Bending Moment, Slope and Deflection of the Beam object.

Parameters

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

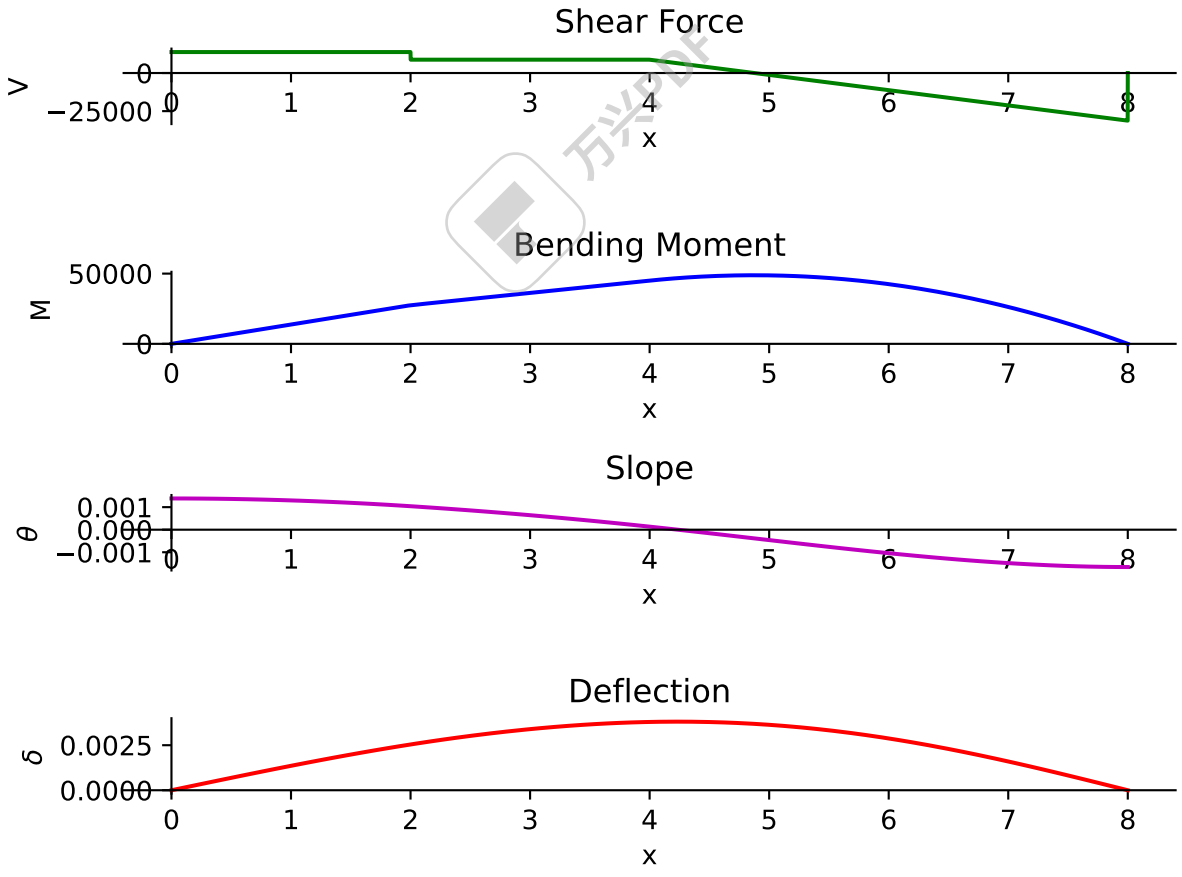

I.L.D. for Shear
V
X
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12

Examples

There is a beam of length 8 meters. A constant distributed load of 10 KN/m is applied from half of the beam till the end. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. A pointload of magnitude 5 KN is also applied from top of the beam, at a distance of 4 meters from the starting point. Take E = 200 GPa and I = 400*(10**-6) meter**4.

Using the sign convention of downwards forces being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(8, 200*(10**9), 400*(10**-6))
>>> b.apply_load(5000, 2, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 8, -1)
>>> b.apply_load(10000, 4, 0, end=8)
>>> b.bc_deflection = [(0, 0), (8, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> axes = b.plot_loading_results()
```



plot_shear_force(*subs=None*)

Returns a plot for Shear force present in the Beam object.

Parameters
subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 8 meters. A constant distributed load of 10 KN/m is applied from half of the beam till the end. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. A pointload of magnitude 5 KN is also applied from top of the beam, at a distance of 4 meters from the starting point. Take E = 200 GPa and I = 400*(10**-6) meter**4.

Using the sign convention of downwards forces being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(8, 200*(10**9), 400*(10**-6))
>>> b.apply_load(5000, 2, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 8, -1)
>>> b.apply_load(10000, 4, 0, end=8)
>>> b.bc_deflection = [(0, 0), (8, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.plot_shear_force()
Plot object containing:
[0]: cartesian line: 13750*SingularityFunction(x, 0, 0) -␣
 ↪5000*SingularityFunction(x, 2, 0)
- 10000*SingularityFunction(x, 4, 1) + 31250*SingularityFunction(x, 8,
 ↪ 0)
+ 10000*SingularityFunction(x, 8, 1) for x over (0.0, 8.0)
```

plot_shear_stress(*subs=None*)

Returns a plot of shear stress present in the beam object.

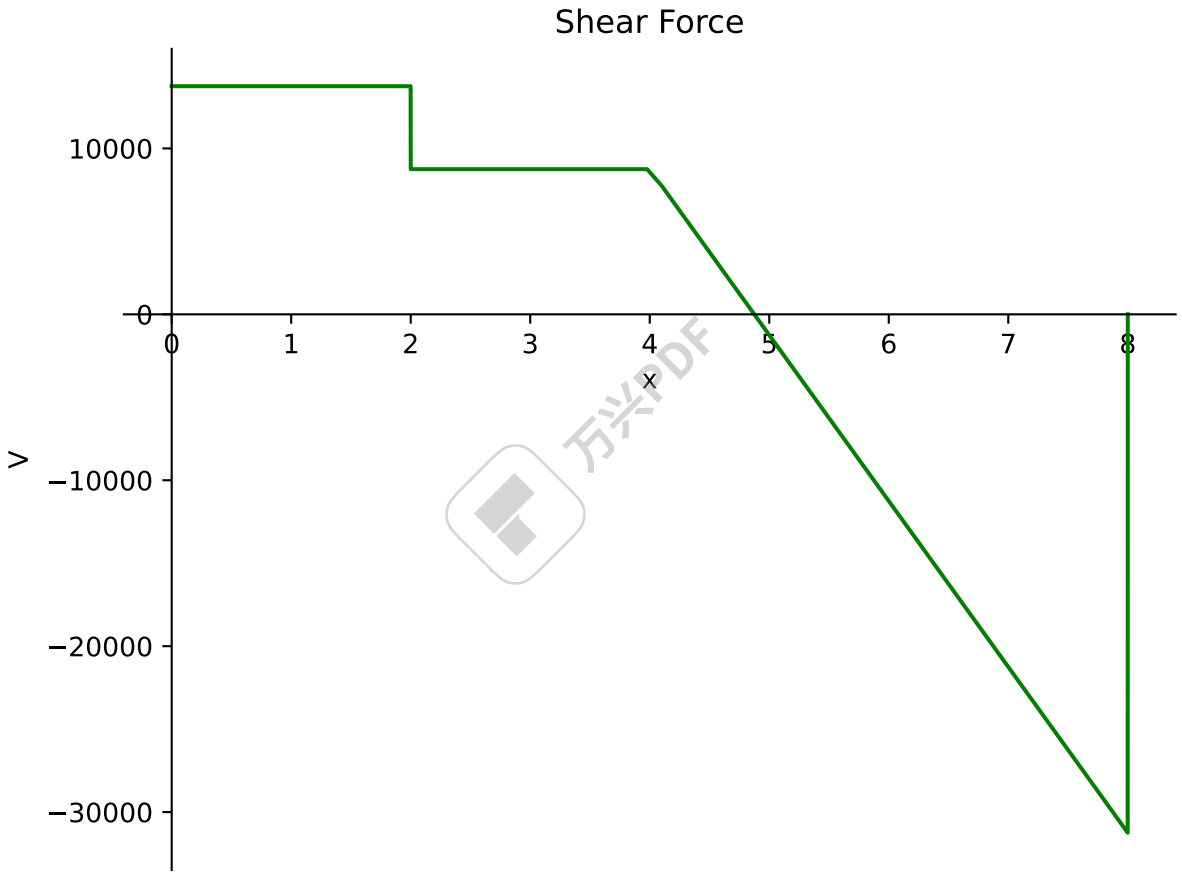
Parameters
subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

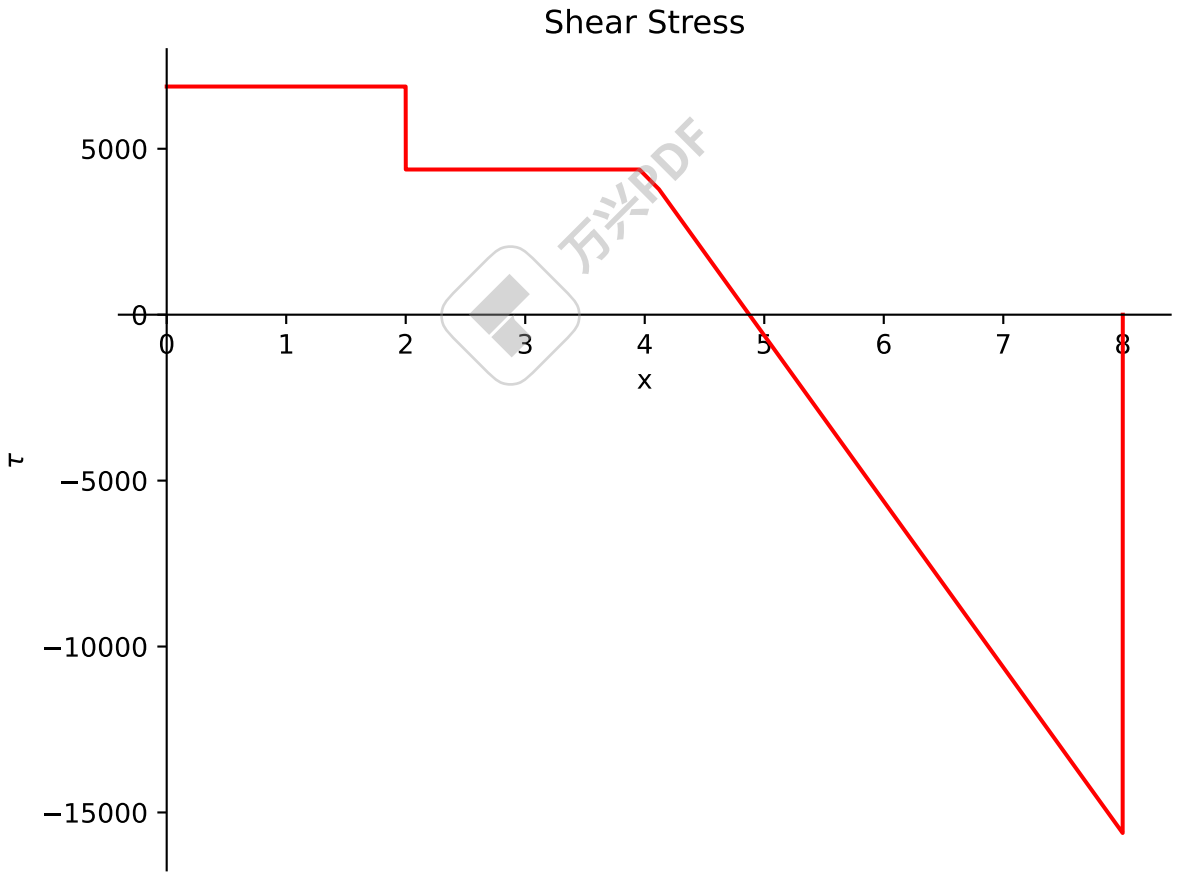
Examples

There is a beam of length 8 meters and area of cross section 2 square meters. A constant distributed load of 10 KN/m is applied from half of the beam till the end. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. A pointload of magnitude 5 KN is also applied from top of the beam, at a distance of 4 meters from the starting point. Take E = 200 GPa and I = 400*(10**-6) meter**4.

Using the sign convention of downwards forces being positive.


Shear Force
10000
0
−10000
−20000
−30000
V
0
1
2
3
4
5
6
7
8
x

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(8, 200*(10**9), 400*(10**-6), 2)
>>> b.apply_load(5000, 2, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 8, -1)
>>> b.apply_load(10000, 4, 0, end=8)
>>> b.bc_deflection = [(0, 0), (8, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.plot_shear_stress()
Plot object containing:
[0]: cartesian line: 6875*SingularityFunction(x, 0, 0) -␣
↪2500*SingularityFunction(x, 2, 0)
- 5000*SingularityFunction(x, 4, 1) + 15625*SingularityFunction(x, 8,␣
↪0)
+ 5000*SingularityFunction(x, 8, 1) for x over (0.0, 8.0)
```



plot_slope(*subs=None*)

Returns a plot for slope of deflection curve of the Beam object.

Parameters

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 8 meters. A constant distributed load of 10 KN/m is applied from half of the beam till the end. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. A pointload of magnitude 5 KN is also applied from top of the beam, at a distance of 4 meters from the starting point. Take E = 200 GPa and I = 400*(10**-6) meter**4.

Using the sign convention of downwards forces being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(8, 200*(10**9), 400*(10**-6))
>>> b.apply_load(5000, 2, -1)
>>> b.apply_load(R1, 0, -1)
>>> b.apply_load(R2, 8, -1)
>>> b.apply_load(10000, 4, 0, end=8)
>>> b.bc_deflection = [(0, 0), (8, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.plot_slope()
Plot object containing:
[0]: cartesian line: -8.59375e-5*SingularityFunction(x, 0, 2) + 3.
 ↪125e-5*SingularityFunction(x, 2, 2)
+ 2.08333333333333e-5*SingularityFunction(x, 4, 3) - 0.
 ↪0001953125*SingularityFunction(x, 8, 2)
- 2.08333333333333e-5*SingularityFunction(x, 8, 3) + 0.
 ↪0013854166666667 for x over (0.0, 8.0)
```

point_cflexure()

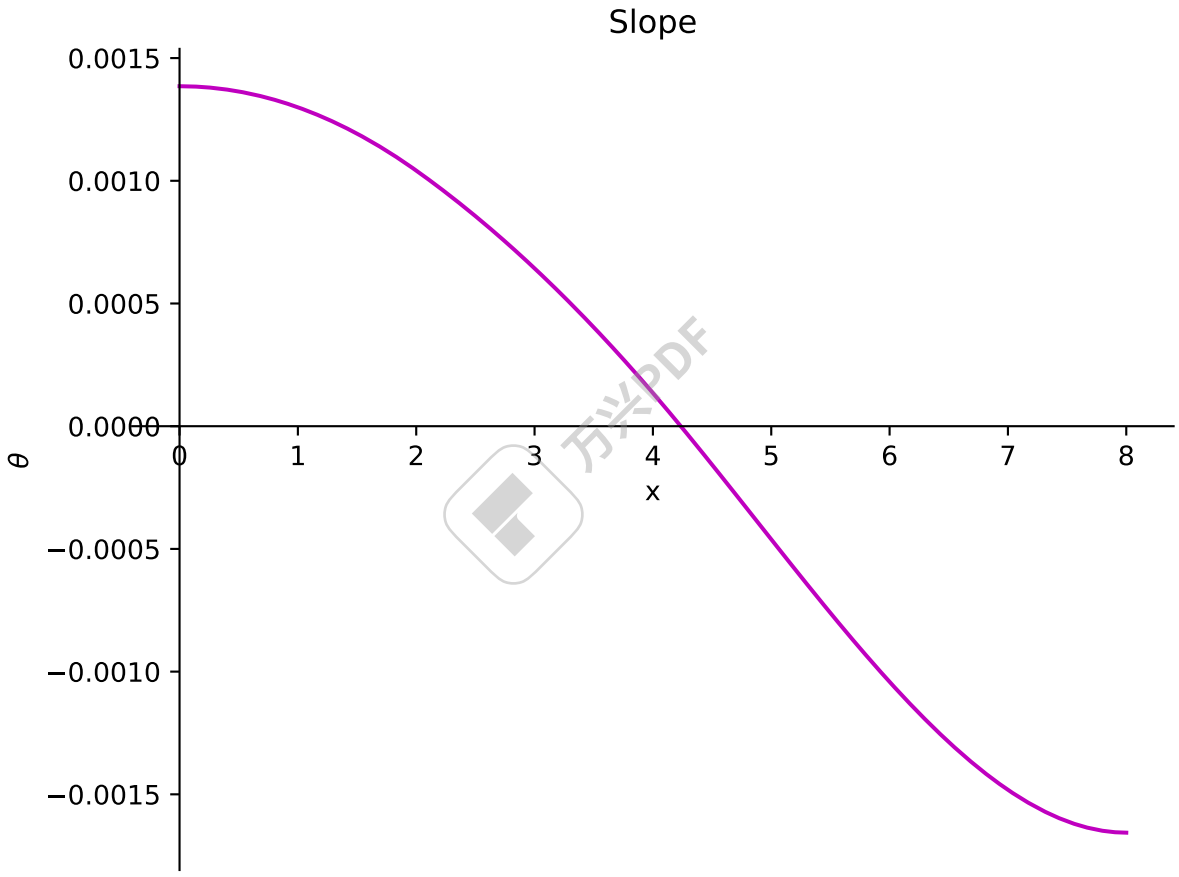
Returns a Set of point(s) with zero bending moment and where bending moment curve of the beam object changes its sign from negative to positive or vice versa.

Examples

There is is 10 meter long overhanging beam. There are two simple supports below the beam. One at the start and another one at a distance of 6 meters from the start. Point loads of magnitude 10KN and 20KN are applied at 2 meters and 4 meters from start respectively. A Uniformly distribute load of magnitude of magnitude 3KN/m is also applied on top starting from 6 meters away from starting point till end. Using the sign convention of upward forces and clockwise moment being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
>>> b = Beam(10, E, I)
>>> b.apply_load(-4, 0, -1)
>>> b.apply_load(-46, 6, -1)
>>> b.apply_load(10, 2, -1)
```

(continues on next page)


Slope
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
θ
0
1
2
3
4
5
6
7
8
x

(continued from previous page)

```
>>> b.apply_load(20, 4, -1)
>>> b.apply_load(3, 6, 0)
>>> b.point_cflexure()
[10/3]
```

property reaction_loads

Returns the reaction forces in a dictionary.

remove_load(*value, start, order, end=None*)

This method removes a particular load present on the beam object. Returns a ValueError if the load passed as an argument is not present on the beam.

Parameters

value : Sympifyable

The magnitude of an applied load.

start : Sympifyable

The starting point of the applied load. For point moments and point forces this is the location of application.

order : Integer

The order of the applied load. - For moments, order= -2 - For point loads, order=-1 - For constant distributed load, order=0 - For ramp loads, order=1 - For parabolic ramp loads, order=2 - ... so on.

end : Sympifyable, optional

An optional argument that can be used if the load has an end point within the length of the beam.

Examples

There is a beam of length 4 meters. A moment of magnitude 3 Nm is applied in the clockwise direction at the starting point of the beam. A pointload of magnitude 4 N is applied from the top of the beam at 2 meters from the starting point and a parabolic ramp load of magnitude 2 N/m is applied below the beam starting from 2 meters to 3 meters away from the starting point of the beam.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
>>> b = Beam(4, E, I)
>>> b.apply_load(-3, 0, -2)
>>> b.apply_load(4, 2, -1)
>>> b.apply_load(-2, 2, 2, end=3)
>>> b.load
-3*SingularityFunction(x, 0, -2) + 4*SingularityFunction(x, 2, -1) -␣
↪2*SingularityFunction(x, 2, 2) + 2*SingularityFunction(x, 3, 0) +␣
↪4*SingularityFunction(x, 3, 1) + 2*SingularityFunction(x, 3, 2)
>>> b.remove_load(-2, 2, 2, end = 3)
>>> b.load
-3*SingularityFunction(x, 0, -2) + 4*SingularityFunction(x, 2, -1)
```

property second_moment

Second moment of area of the Beam.

shear_force()

Returns a Singularity Function expression which represents the shear force curve of the Beam object.

Examples

There is a beam of length 30 meters. A moment of magnitude 120 Nm is applied in the clockwise direction at the end of the beam. A pointload of magnitude 8 N is applied from the top of the beam at the starting point. There are two simple supports below the beam. One at the end and another one at a distance of 10 meters from the start. The deflection is restricted at both the supports.

Using the sign convention of upward forces and clockwise moment being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(30, E, I)
>>> b.apply_load(-8, 0, -1)
>>> b.apply_load(R1, 10, -1)
>>> b.apply_load(R2, 30, -1)
>>> b.apply_load(120, 30, -2)
>>> b.bc_deflection = [(10, 0), (30, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.shear_force()
8*SingularityFunction(x, 0, 0) - 6*SingularityFunction(x, 10, 0) -␣
↪120*SingularityFunction(x, 30, -1) - 2*SingularityFunction(x, 30, 0)
```

shear_stress()

Returns an expression representing the Shear Stress curve of the Beam object.

slope()

Returns a Singularity Function expression which represents the slope the elastic curve of the Beam object.

Examples

There is a beam of length 30 meters. A moment of magnitude 120 Nm is applied in the clockwise direction at the end of the beam. A pointload of magnitude 8 N is applied from the top of the beam at the starting point. There are two simple supports below the beam. One at the end and another one at a distance of 10 meters from the start. The deflection is restricted at both the supports.

Using the sign convention of upward forces and clockwise moment being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
```

(continues on next page)

(continued from previous page)

```
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(30, E, I)
>>> b.apply_load(-8, 0, -1)
>>> b.apply_load(R1, 10, -1)
>>> b.apply_load(R2, 30, -1)
>>> b.apply_load(120, 30, -2)
>>> b.bc_deflection = [(10, 0), (30, 0)]
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.slope()
(-4*SingularityFunction(x, 0, 2) + 3*SingularityFunction(x, 10, 2)
    + 120*SingularityFunction(x, 30, 1) + SingularityFunction(x, 30,␣
↪2) + 4000/3)/(E*I)
```

solve_for_ild_moment(*distance*, *value*, **reactions*)

Determines the Influence Line Diagram equations for moment at a specified point under the effect of a moving load.

Parameters

distance : Integer

Distance of the point from the start of the beam for which equations are to be determined

value : Integer

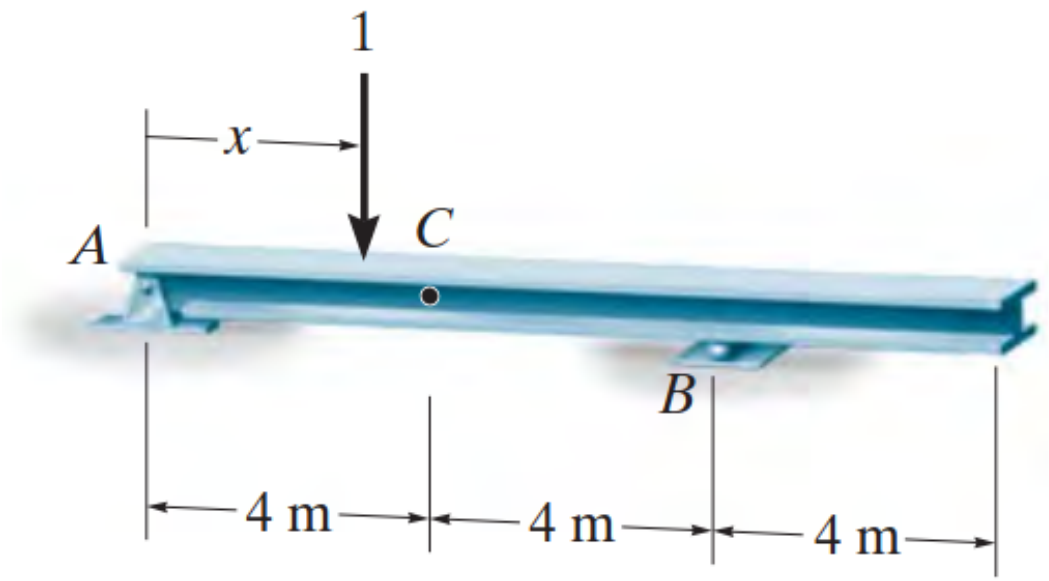
Magnitude of moving load

reactions :

The reaction forces applied on the beam.

Examples

There is a beam of length 12 meters. There are two simple supports below the beam, one at the starting point and another at a distance of 8 meters. Calculate the I.L.D. equations for Moment at a distance of 4 meters under the effect of a moving load of magnitude 1kN.



Using the sign convention of downwards forces being positive.

```
>>> from sympy import symbols
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> E, I = symbols('E, I')
>>> R_0, R_8 = symbols('R_0, R_8')
>>> b = Beam(12, E, I)
>>> b.apply_support(0, 'roller')
>>> b.apply_support(8, 'roller')
>>> b.solve_for_ild_reactions(1, R_0, R_8)
>>> b.solve_for_ild_moment(4, 1, R_0, R_8)
>>> b.ild_moment
Piecewise((-x/2, x < 4), (x/2 - 4, x > 4))
```

solve_for_ild_reactions(*value*, **reactions*)

Determines the Influence Line Diagram equations for reaction forces under the effect of a moving load.

Parameters

value : Integer

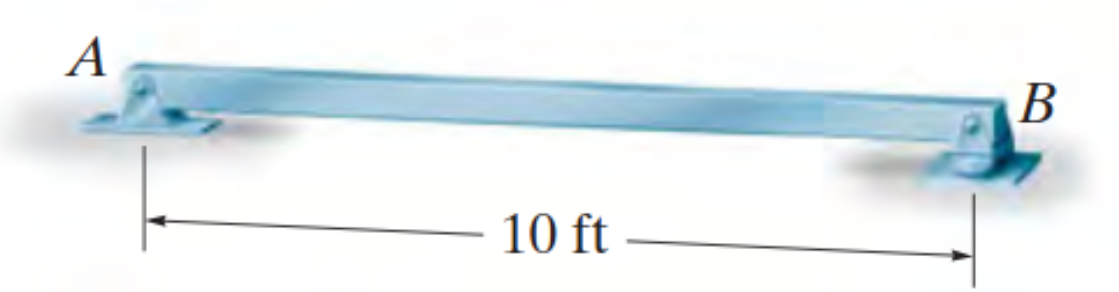
Magnitude of moving load

reactions :

The reaction forces applied on the beam.

Examples

There is a beam of length 10 meters. There are two simple supports below the beam, one at the starting point and another at the ending point of the beam. Calculate the I.L.D. equations for reaction forces under the effect of a moving load of magnitude 1kN.



Using the sign convention of downwards forces being positive.

```
>>> from sympy import symbols
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> E, I = symbols('E, I')
>>> R_0, R_10 = symbols('R_0, R_10')
>>> b = Beam(10, E, I)
>>> b.apply_support(0, 'roller')
>>> b.apply_support(10, 'roller')
>>> b.solve_for_ild_reactions(1,R_0,R_10)
>>> b.ild_reactions
{R_0: x/10 - 1, R_10: -x/10}
```

solve_for_ild_shear(*distance*, *value*, **reactions*)

Determines the Influence Line Diagram equations for shear at a specified point under the effect of a moving load.

Parameters

distance : Integer

Distance of the point from the start of the beam for which equations are to be determined

value : Integer

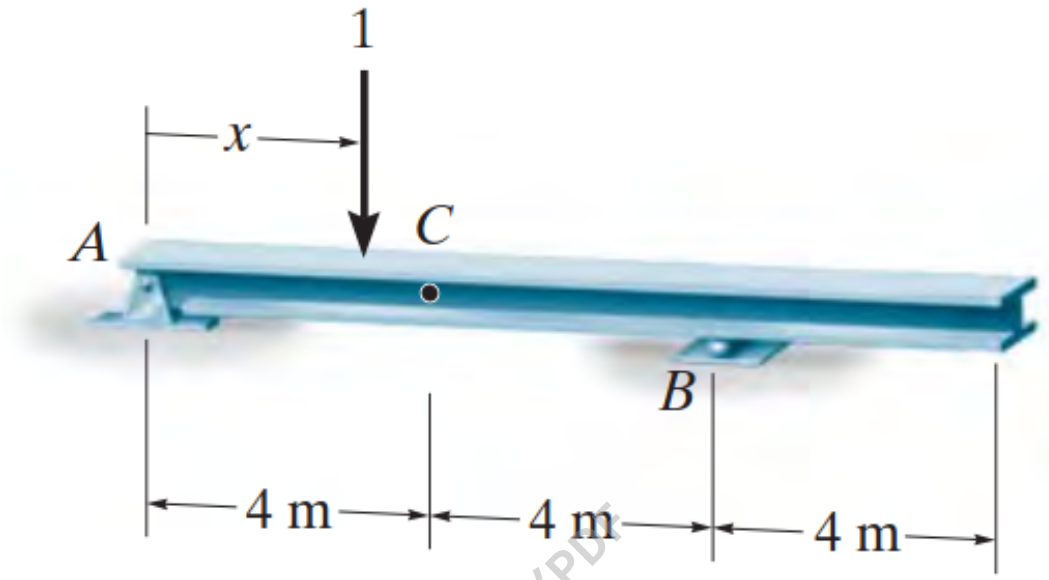
Magnitude of moving load

reactions :

The reaction forces applied on the beam.

Examples

There is a beam of length 12 meters. There are two simple supports below the beam, one at the starting point and another at a distance of 8 meters. Calculate the I.L.D. equations for Shear at a distance of 4 meters under the effect of a moving load of magnitude 1kN.



Using the sign convention of downwards forces being positive.

```
>>> from sympy import symbols
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> E, I = symbols('E, I')
>>> R_0, R_8 = symbols('R_0, R_8')
>>> b = Beam(12, E, I)
>>> b.apply_support(0, 'roller')
>>> b.apply_support(8, 'roller')
>>> b.solve_for_ild_reactions(1, R_0, R_8)
>>> b.solve_for_ild_shear(4, 1, R_0, R_8)
>>> b.ild_shear
Piecewise((x/8, x < 4), (x/8 - 1, x > 4))
```

solve_for_reaction_loads(**reactions*)

Solves for the reaction forces.

Examples

There is a beam of length 30 meters. A moment of magnitude 120 Nm is applied in the clockwise direction at the end of the beam. A pointload of magnitude 8 N is applied from the top of the beam at the starting point. There are two simple supports below the beam. One at the end and another one at a distance of 10 meters from the start. The deflection is restricted at both the supports.

Using the sign convention of upward forces and clockwise moment being positive.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I = symbols('E, I')
>>> R1, R2 = symbols('R1, R2')
>>> b = Beam(30, E, I)
>>> b.apply_load(-8, 0, -1)
>>> b.apply_load(R1, 10, -1)  # Reaction force at x = 10
>>> b.apply_load(R2, 30, -1)  # Reaction force at x = 30
>>> b.apply_load(120, 30, -2)
>>> b.bc_deflection = [(10, 0), (30, 0)]
>>> b.load
R1*SingularityFunction(x, 10, -1) + R2*SingularityFunction(x, 30, -1)
    - 8*SingularityFunction(x, 0, -1) + 120*SingularityFunction(x, 30,
 ↪ -2)
>>> b.solve_for_reaction_loads(R1, R2)
>>> b.reaction_loads
{R1: 6, R2: 2}
>>> b.load
-8*SingularityFunction(x, 0, -1) + 6*SingularityFunction(x, 10, -1)
    + 120*SingularityFunction(x, 30, -2) + 2*SingularityFunction(x,␣
 ↪30, -1)
```

property variable

A symbol that can be used as a variable along the length of the beam while representing load distribution, shear force curve, bending moment, slope curve and the deflection curve. By default, it is set to `Symbol('x')`, but this property is mutable.

Examples

```
>>> from sympy.physics.continuum_mechanics.beam import Beam
>>> from sympy import symbols
>>> E, I, A = symbols('E, I, A')
>>> x, y, z = symbols('x, y, z')
>>> b = Beam(4, E, I)
>>> b.variable
x
>>> b.variable = y
>>> b.variable
y
>>> b = Beam(4, E, I, A, z)
>>> b.variable
z
```

class sympy.physics.continuum_mechanics.beam.**Beam3D**(*length*, *elastic_modulus*, *shear_modulus*, *second_moment*, *area*, *variable=x*)

This class handles loads applied in any direction of a 3D space along with unequal values of Second moment along different axes.

Note: A consistent sign convention must be used while solving a beam bending problem; the results will automatically follow the chosen sign convention. This class assumes that any kind of distributed load/moment is applied through out the span of a beam.

Examples

There is a beam of l meters long. A constant distributed load of magnitude q is applied along y-axis from start till the end of beam. A constant distributed moment of magnitude m is also applied along z-axis from start till the end of beam. Beam is fixed at both of its end. So, deflection of the beam at the both ends is restricted.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols, simplify, collect, factor
>>> l, E, G, I, A = symbols('l, E, G, I, A')
>>> b = Beam3D(l, E, G, I, A)
>>> x, q, m = symbols('x, q, m')
>>> b.apply_load(q, 0, 0, dir="y")
>>> b.apply_moment_load(m, 0, -1, dir="z")
>>> b.shear_force()
[0, -q*x, 0]
>>> b.bending_moment()
[0, 0, -m*x + q*x**2/2]
>>> b.bc_slope = [(0, [0, 0, 0]), (l, [0, 0, 0])]
>>> b.bc_deflection = [(0, [0, 0, 0]), (l, [0, 0, 0])]
>>> b.solve_slope_deflection()
>>> factor(b.slope())
[0, 0, x*(-l + x)*(-A*G*l**3*q + 2*A*G*l**2*q*x - 12*E*I*l*q
    - 72*E*I*m + 24*E*I*q*x)/(12*E*I*(A*G*l**2 + 12*E*I))]
>>> dx, dy, dz = b.deflection()
>>> dy = collect(simplify(dy), x)
>>> dx == dz == 0
True
>>> dy == (x*(12*E*I*l*(A*G*l**2*q - 2*A*G*l*m + 12*E*I*q)
... + x*(A*G*l*(3*l*(A*G*l**2*q - 2*A*G*l*m + 12*E*I*q) + x*(-
↪2*A*G*l**2*q + 4*A*G*l*m - 24*E*I*q))
... + A*G*(A*G*l**2 + 12*E*I)*(-2*l**2*q + 6*l*m - 4*m*x + q*x**2)
... - 12*E*I*q*(A*G*l**2 + 12*E*I)))/(24*A*E*G*I*(A*G*l**2 + 12*E*I)))
True
```

References

[R651]

apply_load(*value*, *start*, *order*, *dir*='*y*')

This method adds up the force load to a particular beam object.

Parameters

value : Sympifyable

The magnitude of an applied load.

dir : String

Axis along which load is applied.

order : Integer

The order of the applied load. - For point loads, order=-1 - For constant distributed load, order=0 - For ramp loads, order=1 - For parabolic ramp loads, order=2 - ... so on.

apply_moment_load(*value*, *start*, *order*, *dir*='*y*')

This method adds up the moment loads to a particular beam object.

Parameters

value : Sympifyable

The magnitude of an applied moment.

dir : String

Axis along which moment is applied.

order : Integer

The order of the applied load. - For point moments, order=-2 - For constant distributed moment, order=-1 - For ramp moments, order=0 - For parabolic ramp moments, order=1 - ... so on.

property area

Cross-sectional area of the Beam.

axial_force()

Returns expression of Axial shear force present inside the Beam object.

axial_stress()

Returns expression of Axial stress present inside the Beam object.

bending_moment()

Returns a list of three expressions which represents the bending moment curve of the Beam object along all three axes.

property boundary_conditions

Returns a dictionary of boundary conditions applied on the beam. The dictionary has two keywords namely slope and deflection. The value of each keyword is a list of tuple, where each tuple contains location and value of a boundary condition in the format (location, value). Further each value is a list corresponding to slope or deflection(s) values along three axes at that location.

Examples

There is a beam of length 4 meters. The slope at 0 should be 4 along the x-axis and 0 along others. At the other end of beam, deflection along all the three axes should be zero.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(30, E, G, I, A, x)
>>> b.bc_slope = [(0, (4, 0, 0))]
>>> b.bc_deflection = [(4, [0, 0, 0])]
>>> b.boundary_conditions
{'deflection': [(4, [0, 0, 0])], 'slope': [(0, (4, 0, 0))]}
```

Here the deflection of the beam should be `0` along all the three axes at `4`. Similarly, the slope of the beam should be 4 along x-axis and `0` along y and z axis at `0`.

deflection()

Returns a three element list representing deflection curve along all the three axes.

property load_vector

Returns a three element list representing the load vector.

max_bending_moment()

Returns point of max bending moment and its corresponding bending moment value along all directions in a Beam object as a list. solve_for_reaction_loads() must be called before using this function.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, 40, 21, 100, 25, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.max_bending_moment()
[(0, 0), (20, 3000), (20, 16000)]
```

max_bmoment()

Returns point of max bending moment and its corresponding bending moment value

along all directions in a Beam object as a list. solve_for_reaction_loads() must be called before using this function.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, 40, 21, 100, 25, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.max_bending_moment()
[(0, 0), (20, 3000), (20, 16000)]
```

max_deflection()

Returns point of max deflection and its corresponding deflection value along all directions in a Beam object as a list. solve_for_reaction_loads() and solve_slope_deflection() must be called before using this function.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, 40, 21, 100, 25, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.solve_slope_deflection()
>>> b.max_deflection()
```

(continues on next page)

(continued from previous page)

```
[(0, 0), (10, 495/14), (-10 + 10*sqrt(10793)/43, (10 - 10*sqrt(10793)/
↪43)**3/160 - 20/7 + (10 - 10*sqrt(10793)/43)**4/6400 +␣
↪20*sqrt(10793)/301 + 27*(10 - 10*sqrt(10793)/43)**2/560)]
```

max_shear_force()

Returns point of max shear force and its corresponding shear value along all directions in a Beam object as a list. solve_for_reaction_loads() must be called before using this function.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, 40, 21, 100, 25, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.max_shear_force()
[(0, 0), (20, 2400), (20, 300)]
```

property moment_load_vector

Returns a three element list representing moment loads on Beam.

plot_bending_moment(*dir='all', subs=None*)

Returns a plot for bending moment along all three directions present in the Beam object.

Parameters

dir : string (default

Direction along which bending moment plot is required. If no direction is specified, all plots are displayed.

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, E, G, I, A, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.plot_bending_moment()
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: 0 for x over (0.0, 20.0)
Plot[1]:Plot object containing:
[0]: cartesian line: -15*x**2/2 for x over (0.0, 20.0)
Plot[2]:Plot object containing:
[0]: cartesian line: 2*x**3 for x over (0.0, 20.0)
```

plot_deflection(*dir='all'*, *subs=None*)

Returns a plot for Deflection along all three directions present in the Beam object.

Parameters

dir : string (default

Direction along which deflection plot is required. If no direction is specified, all plots are displayed.

subs : dictionary

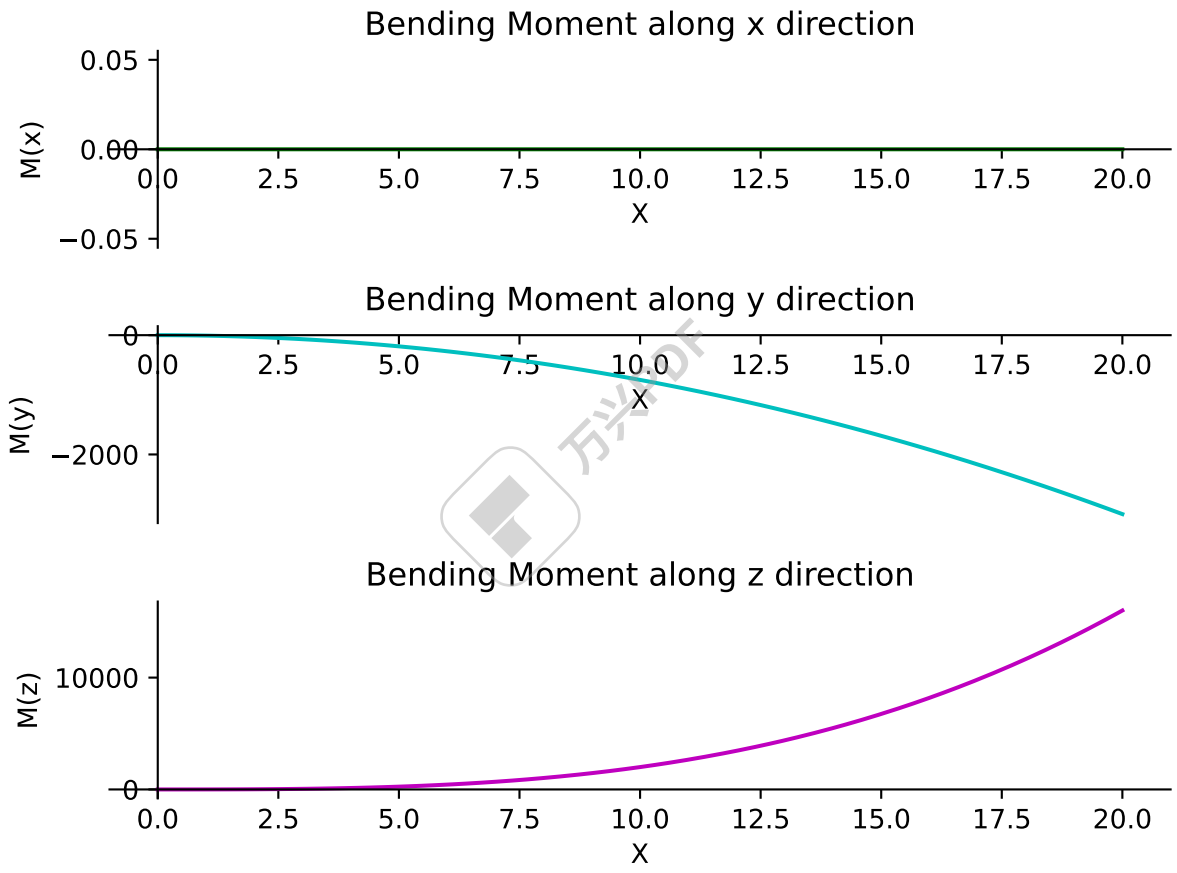
Python dictionary containing Symbols as keys and their corresponding values.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

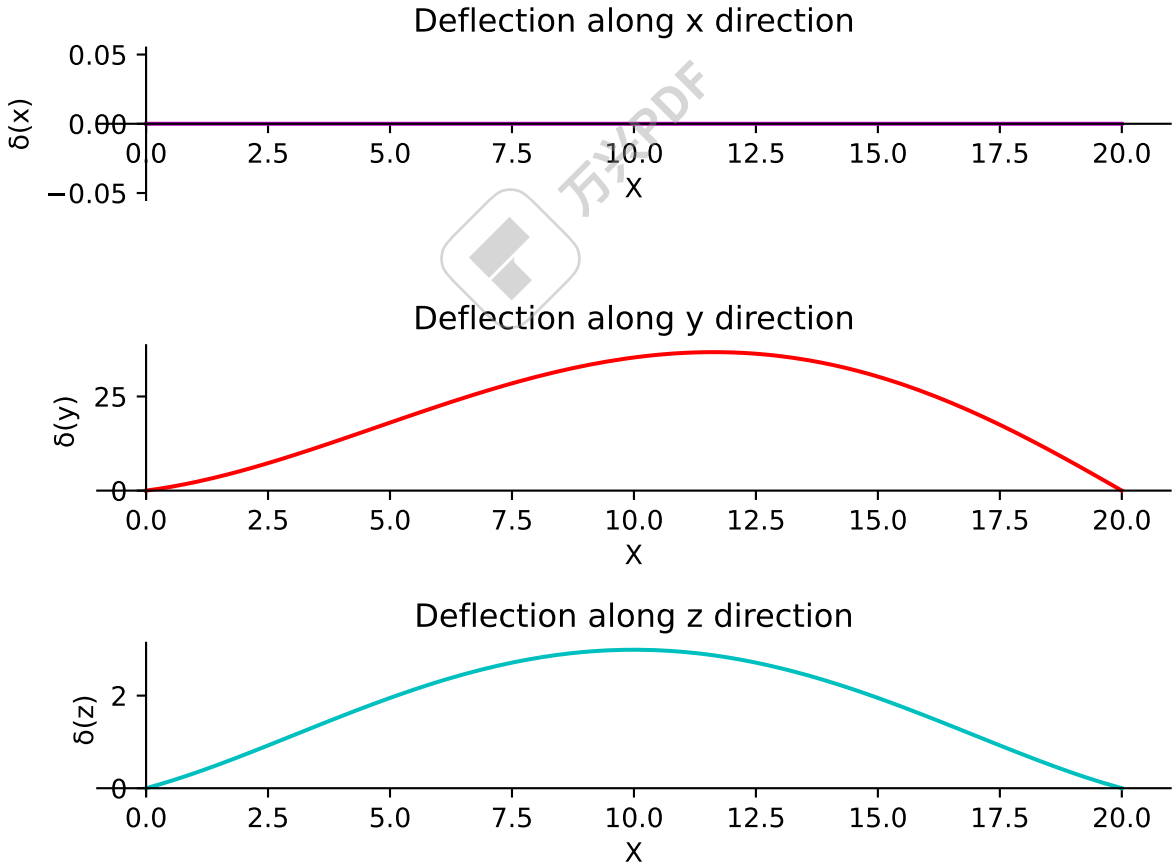
```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, 40, 21, 100, 25, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
```

(continues on next page)


Bending Moment along x direction
M(x)
0.05
0.00
−0.05
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Bending Moment along y direction
M(y)
0
−2000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Bending Moment along z direction
M(z)
10000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X

(continued from previous page)

```
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.solve_slope_deflection()
>>> b.plot_deflection()
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: 0 for x over (0.0, 20.0)
Plot[1]:Plot object containing:
[0]: cartesian line: x**5/40000 - 4013*x**3/90300 + 26*x**2/43 +␣
↪1520*x/903 for x over (0.0, 20.0)
Plot[2]:Plot object containing:
[0]: cartesian line: x**4/6400 - x**3/160 + 27*x**2/560 + 2*x/7 for x␣
↪over (0.0, 20.0)
```



plot_loading_results(*dir='x'*, *subs=None*)

Returns a subplot of Shear Force, Bending Moment, Slope and Deflection of the Beam object along the direction specified.

Parameters

dir : string (default

Direction along which plots are required. If no direction is specified, plots along x-axis are displayed.

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, E, G, I, A, x)
>>> subs = {E:40, G:21, I:100, A:25}
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.solve_slope_deflection()
>>> b.plot_loading_results('y',subs)
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: -6*x**2 for x over (0.0, 20.0)
Plot[1]:Plot object containing:
[0]: cartesian line: -15*x**2/2 for x over (0.0, 20.0)
Plot[2]:Plot object containing:
[0]: cartesian line: -x**3/1600 + 3*x**2/160 - x/8 for x over (0.0,␣
↪20.0)
Plot[3]:Plot object containing:
[0]: cartesian line: x**5/40000 - 4013*x**3/90300 + 26*x**2/43 +␣
↪1520*x/903 for x over (0.0, 20.0)
```

plot_shear_force(*dir='all'*, *subs=None*)

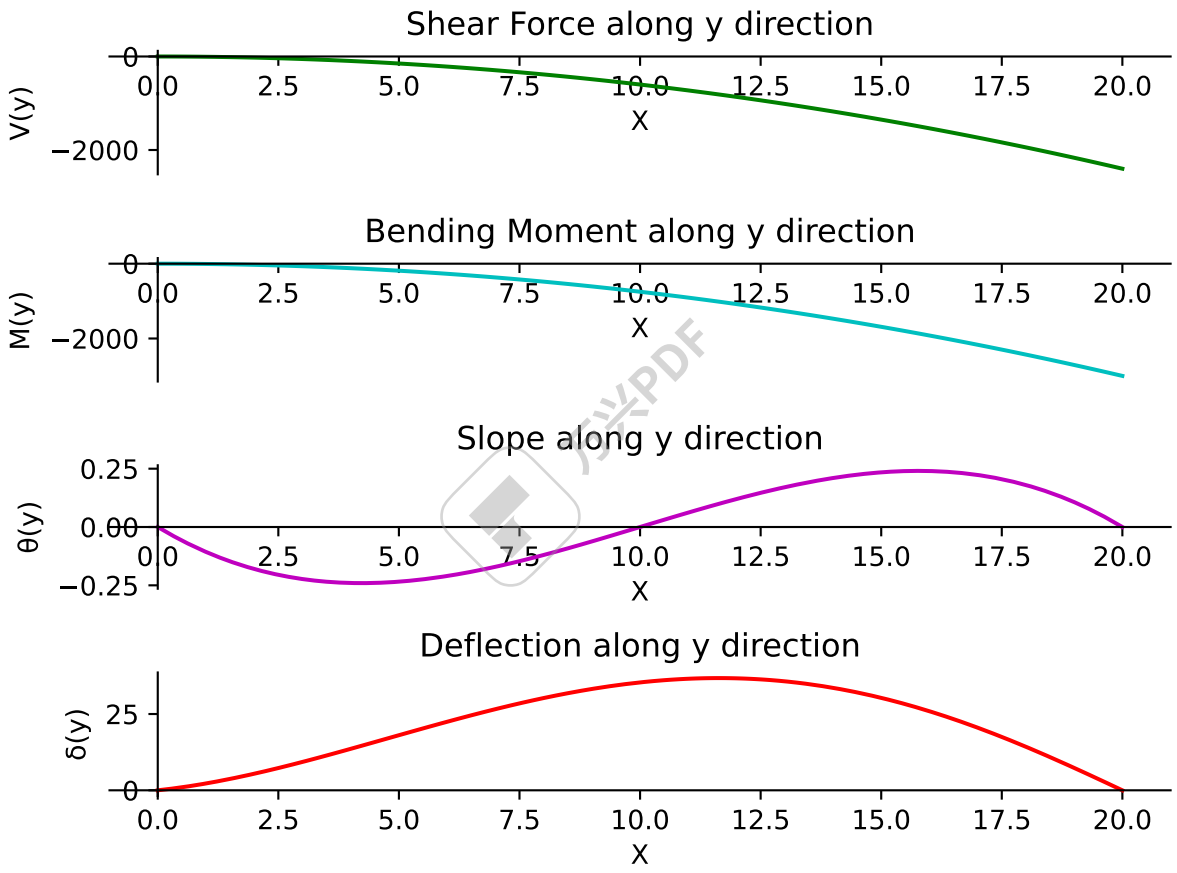
Returns a plot for Shear force along all three directions present in the Beam object.

Parameters

dir : string (default

Direction along which shear force plot is required. If no direction is specified, all plots are displayed.

subs : dictionary


Shear Force along y direction
V(y)
0
−2000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Bending Moment along y direction
M(y)
0
−2000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Slope along y direction
θ(y)
0.25
0.00
−0.25
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Deflection along y direction
δ(y)
25
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X

Python dictionary containing Symbols as key and their corresponding values.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, E, G, I, A, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.plot_shear_force()
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: 0 for x over (0.0, 20.0)
Plot[1]:Plot object containing:
[0]: cartesian line: -6*x**2 for x over (0.0, 20.0)
Plot[2]:Plot object containing:
[0]: cartesian line: -15*x for x over (0.0, 20.0)
```

plot_shear_stress(*dir='all', subs=None*)

Returns a plot for Shear Stress along all three directions present in the Beam object.

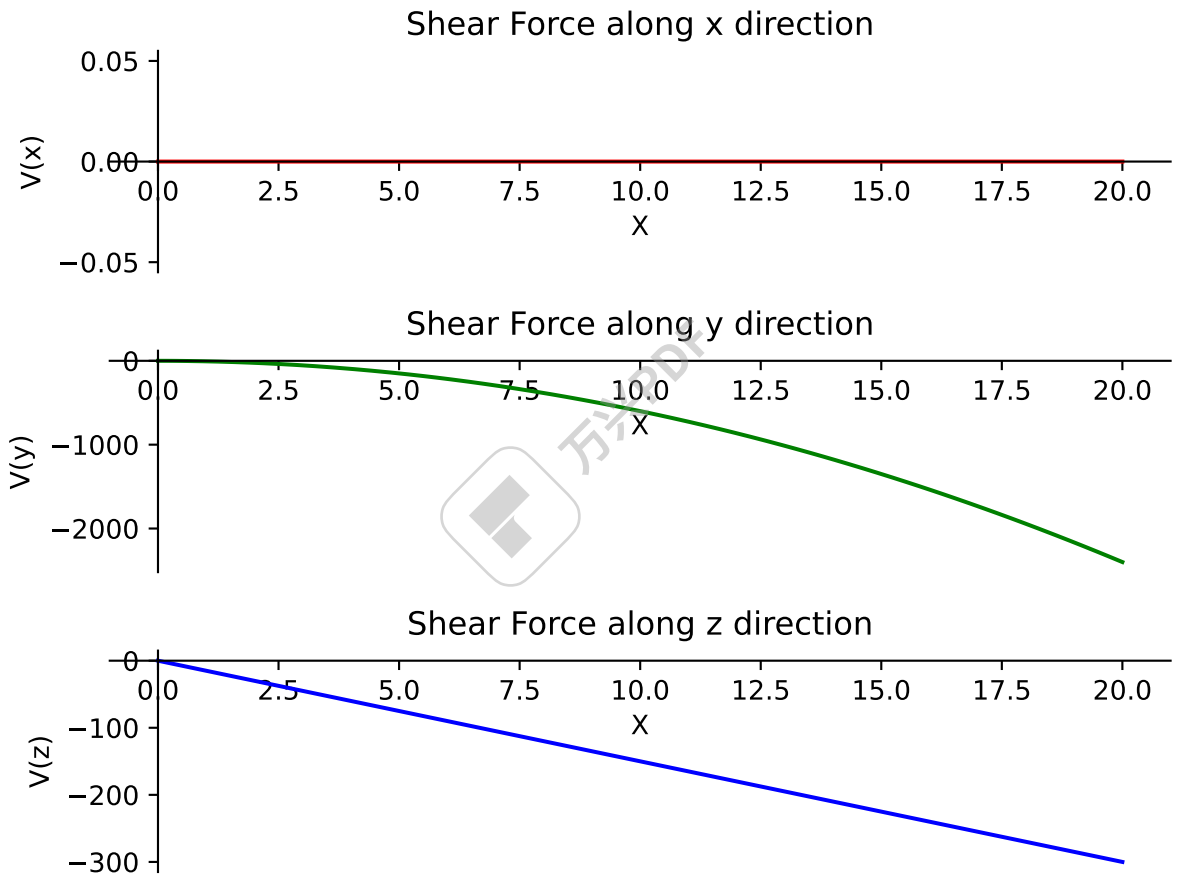
Parameters

dir : string (default

Direction along which shear stress plot is required. If no direction is specified, all plots are displayed.

subs : dictionary

Python dictionary containing Symbols as key and their corresponding values.


Shear Force along x direction
V(x)
0.05
0.00
−0.05
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Shear Force along y direction
V(y)
0
−1000
−2000
Shear Force along z direction
V(z)
−100
−200
−300

Examples

There is a beam of length 20 meters and area of cross section 2 square meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, E, G, I, 2, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.plot_shear_stress()
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: 0 for x over (0.0, 20.0)
Plot[1]:Plot object containing:
[0]: cartesian line: -3*x**2 for x over (0.0, 20.0)
Plot[2]:Plot object containing:
[0]: cartesian line: -15*x/2 for x over (0.0, 20.0)
```

plot_slope(*dir='all'*, *subs=None*)

Returns a plot for Slope along all three directions present in the Beam object.

Parameters

dir : string (default

Direction along which Slope plot is required. If no direction is specified, all plots are displayed.

subs : dictionary

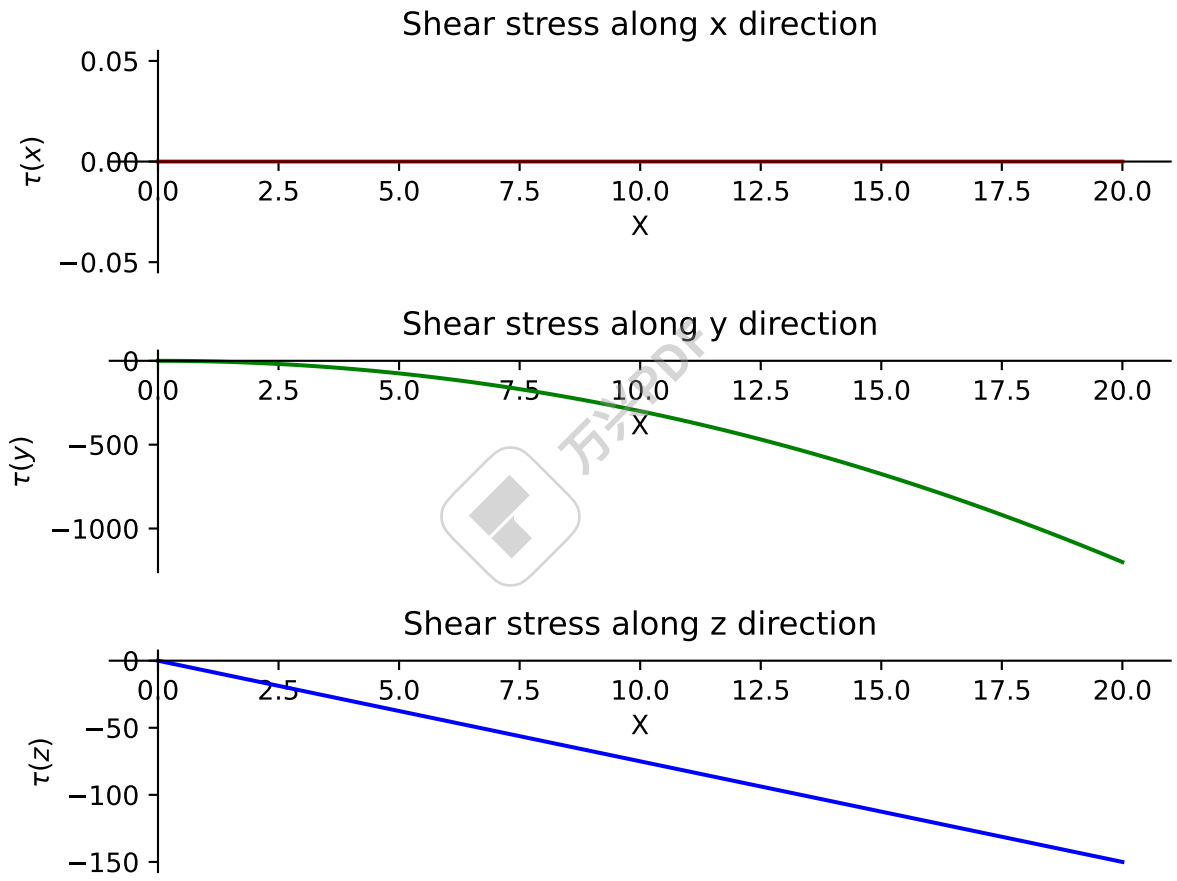
Python dictionary containing Symbols as keys and their corresponding values.

Examples

There is a beam of length 20 meters. It it supported by rollers at of its end. A linear load having slope equal to 12 is applied along y-axis. A constant distributed load of magnitude 15 N is applied from start till its end along z-axis.

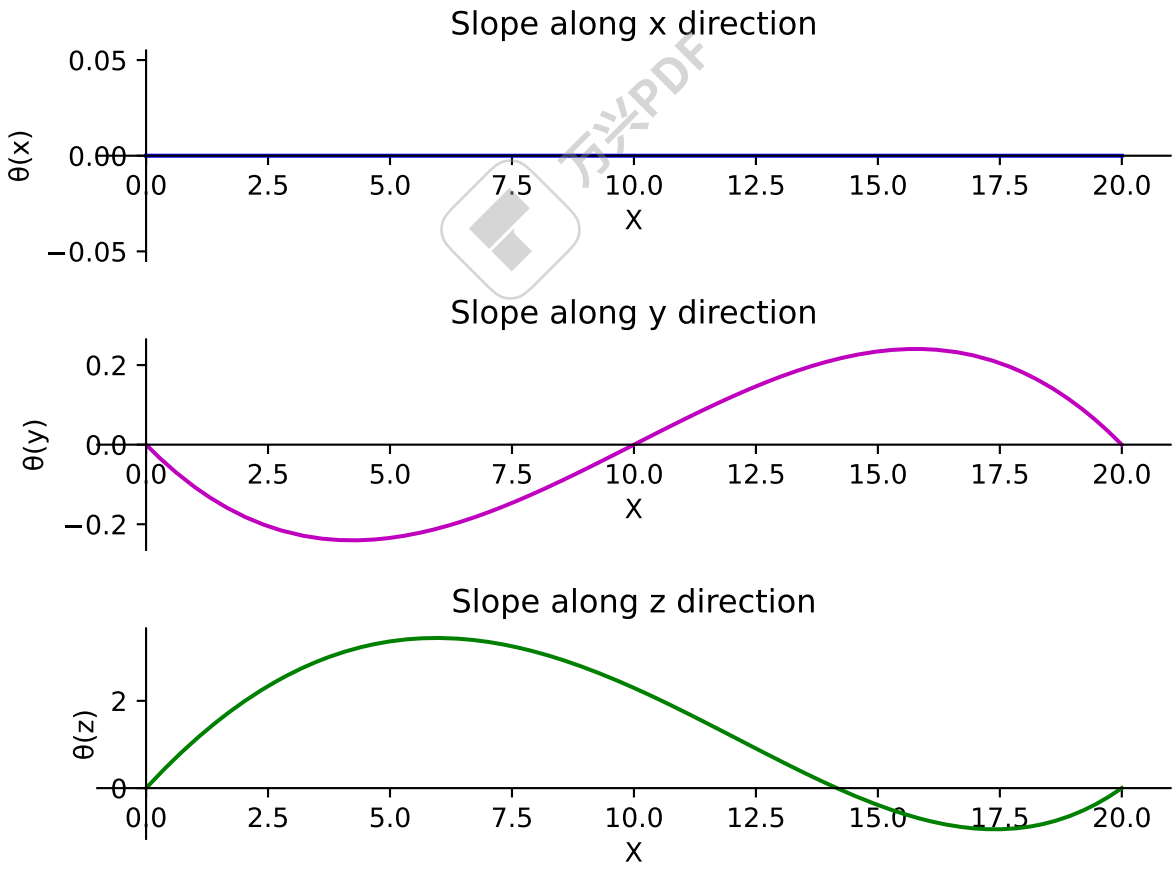
```
>>> from sympy.physics.continuum_mechanics.beam import Beam3D
>>> from sympy import symbols
>>> l, E, G, I, A, x = symbols('l, E, G, I, A, x')
>>> b = Beam3D(20, 40, 21, 100, 25, x)
>>> b.apply_load(15, start=0, order=0, dir="z")
```

(continues on next page)


Shear stress along x direction
0.05
0.00
−0.05
τ(x)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X
Shear stress along y direction
0
−500
−1000
τ(y)
X
Shear stress along z direction
0
−50
−100
−150
τ(z)
X

(continued from previous page)

```
>>> b.apply_load(12*x, start=0, order=0, dir="y")
>>> b.bc_deflection = [(0, [0, 0, 0]), (20, [0, 0, 0])]
>>> R1, R2, R3, R4 = symbols('R1, R2, R3, R4')
>>> b.apply_load(R1, start=0, order=-1, dir="z")
>>> b.apply_load(R2, start=20, order=-1, dir="z")
>>> b.apply_load(R3, start=0, order=-1, dir="y")
>>> b.apply_load(R4, start=20, order=-1, dir="y")
>>> b.solve_for_reaction_loads(R1, R2, R3, R4)
>>> b.solve_slope_deflection()
>>> b.plot_slope()
PlotGrid object containing:
Plot[0]:Plot object containing:
[0]: cartesian line: 0 for x over (0.0, 20.0)
Plot[1]:Plot object containing:
[0]: cartesian line: -x**3/1600 + 3*x**2/160 - x/8 for x over (0.0,␣
↪20.0)
Plot[2]:Plot object containing:
[0]: cartesian line: x**4/8000 - 19*x**2/172 + 52*x/43 for x over (0.
↪0, 20.0)
```



polar_moment()

Returns the polar moment of area of the beam about the X axis with respect to the