Truss Documentation

Release 0.1

Ludovic Charleux

Mar 30, 2017

Contents

Truss is a 2D truss calculation framework. It is fully scriptable and dedicated to teaching activities.

Contents:

CHAPTER 1

Core

Core regroups the general purpose tools of the Truss package.

Nodes

class `truss.core.Node`(*coords=array([0., 0.]), label=None, force=array([0., 0.]), block=array([False, False], dtype=bool), block_side=1*)

Creates a node.

Parameters

- **coords** (*length 2 float array*) – coordinates of the node.
- **force** (*length 2 float array*) – external force applied on the node.
- **block** (*length 2 boolean array*) – bloked degrees of freedom.
- **label** (*string*) – label of the node.

```
>>> from truss.core import Node
>>> A = Node((1.,5.), label = "A", block = (False, True) )
```

`Node.data()`

Returns the data associated with the node as a pandas.Series object.

```
>>> A.data()
block   bx    True
        by    True
coords  x        0
        y        0
disp    ux       0
        uy       0
force   Fx   -2000
        Fy   -1000
label   o        A
dtype: object
```

Bars

class `truss.core.Bar`(*n1*, *n2*, *section=1.0*, *modulus=1.0*, *density=1.0*, *yield_stress=0.001*)

A bar class.

Parameters

- **n1** (`truss.core.Node` instance.) – start node of the bar.
- **n2** (`truss.core.Node` instance.) – end node of the bar.
- **section** (*float*) – cross section of the bar.
- **modulus** (*float*) – Young's modulus of the bar's constitutive material.
- **density** (*float*) – density of the bar's constitutive material.
- **yield_stress** (*float*) – yield_stress of the bar's constitutive material.

```
>>> from truss.core import Node, Bar, Model
>>> m = Model()
>>> A = m.add_node((0., 1.), label = "A")
>>> B = m.add_node((1., 1.), label = "B")
>>> b = Bar(A, B, section = 1., modulus = 210.e9, density = 7800.)
>>> b
<Bar: (0.0, 1.0) -> (1.0, 1.0), S = 1.0, E = 2.1e+11, rho = 7800.0>
```

`Bar.`**`data`**()

Returns the data associated with the bar as a pandas.Series.

```
>>> AB.data()
conn        c1                    A
            c2                    B
direction   dx                    0
            dy                    1
geometry    length                1
            volume           0.0001
props       density            2700
            mass               0.27
            section          0.0001
state       elongation    4.7619e-05
            strain        4.7619e-05
            stress            1e+07
            tension            1000
dtype: object
```

`Bar.`**`length`**(*deformed=False*)

Returns the length of the bar.

Parameters **deformed** (*Bool*) – False for undeformed configuration, True for deformed configuration.

Return type float

`Bar.`**`volume`**()

Returns the (undeformed) volume of the bar.

Return type float

`Bar.mass()`
Returns the mass of the bar.

Return type float

`Bar.direction(deformed=False)`
Returns the unit vector corresponding to the direction of the bar from start to end.

Return type length 2 array

`Bar.normal(deformed=False)`
Returns the unit vector corresponding to the normal direction of the bar.

Return type length 2 array

`Bar.stiffness()`
Returns the stiffness of the bar.

Return type float

`Bar.stiffness_matrix()`
Returns stiffness matrix of the bar.

Return type (4,4) array

```
>>> from truss.core import Node, Bar, Model
>>> m = Model()
>>> B = m.add_node((0., 1.), label = "B")
>>> A = m.add_node((0., 0.), label = "A")
>>> b = Bar(A, B)
>>> b.stiffness_matrix()
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  1.,  0., -1.],
       [ 0.,  0.,  0.,  0.],
       [ 0., -1.,  0.,  1.]])
```

Models

class `truss.core.Model`
Truss model

`Model.data(at='nodes')`
Returns the data associated with the bars/nodes a dataframe.

Inputs:

- at: should be 'nodes' or 'bars'

```
>>> m.data(at = "nodes")
    block        coords            disp              force       label
    bx     by      x  y           ux            uy    Fx    Fy      o
0   True   True      0  0            0            0 -2000 -1000      A
1   True  False      0  1            0   4.7619e-05  1000     0      B
2  False  False      1  0  9.52381e-05  0.000277544  1000  1000      C
3  False  False      1  1            0  0.000277544     0     0      D
```

```
>>> m.data(at = "bars")
    conn    direction           geometry                 props                 ␣
↪       c1 c2        dx        dy   length       volume density      mass␣
↪section
```

```
0    A  B         0         1        1       0.0001    2700      0.27  0.0001
1    A  C         1         0        1       0.0001    2700      0.27  0.0001
2    B  C  0.707107 -0.707107  1.41421  0.000141421    2700  0.381838  0.0001
3    C  D         0         1        1       0.0001    2700      0.27  0.0001
4    B  D         1         0        1       0.0001    2700      0.27  0.0001
         state
    elongation       strain       stress      tension
0   4.7619e-05   4.7619e-05        1e+07         1000
1  9.52381e-05  9.52381e-05        2e+07         2000
2 -9.52381e-05 -6.73435e-05 -1.41421e+07     -1414.21
3 -5.42101e-20 -5.42101e-20 -1.13841e-08 -1.13841e-12
4            0            0            0            0
```

`Model.`**`add_node`**(*node*, **args*, ***kwargs*)

Adds a node to the model. If `node` is an instance of the `truss.core.node`, it is added. If `node` an length 2 array containing, a node instance is created and `node` it is used as coordinates and other arguments can be provided as well.

Parameters **`node`** (array or `truss.core.node` instance.) – node label or node.

```
>>> m = truss.core.Model()
>>> A = m.add_node((1., 3.), label = "A")
<Node A: x = 1.0, y = 3.0>
>>> B = truss.core.Node((4., 5.), label = "B")
>>> m.add_node(B)
<Node B: x = 4.0, y = 5.0>
```

`Model.`**`add_bar`**(**args*, ***kwargs*)

Add a bar to the model.

```
>>> from truss.core import Node, Bar, Model
>>> m = Model()
>>> B = m.add_node((0., 1.), label = "B")
>>> A = m.add_node((0., 0.), label = "A")
>>> b = Bar(A, B)
```

`Model.`**`add_node`**(*node*, **args*, ***kwargs*)

Adds a node to the model. If `node` is an instance of the `truss.core.node`, it is added. If `node` an length 2 array containing, a node instance is created and `node` it is used as coordinates and other arguments can be provided as well.

Parameters **`node`** (array or `truss.core.node` instance.) – node label or node.

```
>>> m = truss.core.Model()
>>> A = m.add_node((1., 3.), label = "A")
<Node A: x = 1.0, y = 3.0>
>>> B = truss.core.Node((4., 5.), label = "B")
>>> m.add_node(B)
<Node B: x = 4.0, y = 5.0>
```

`Model.`**`add_force`**(*node*, *magnitude*)

Adds an external force on an existing node.

Parameters

- **`node`** (`truss.core.Node` instance.) – node on which the force is added.
- **`magnitude`** (*`length 2 array`*.) – force magnitude on both directions

`Model.stiffness_matrix()`
Returns the full assembled stiffness matrix of the model.

`Model.force_vector()`
Returns the full force vector applied on the system.

`Model.solve()`
Solves the system.

`Model.active_dof()`
Returns the indices of the active (i. e. not blocked) degrees of freedom.

`Model.bbox(deformed=True, factor=0.2)`
Returns the bounding box of the truss.

`Model.draw(ax, deformed=True, field='stress', label=True, forces=True, displacements=False, force_scale=1.0, displacement_scale=1.0)`
Draws the truss in `matplotlib` axes.

Parameters

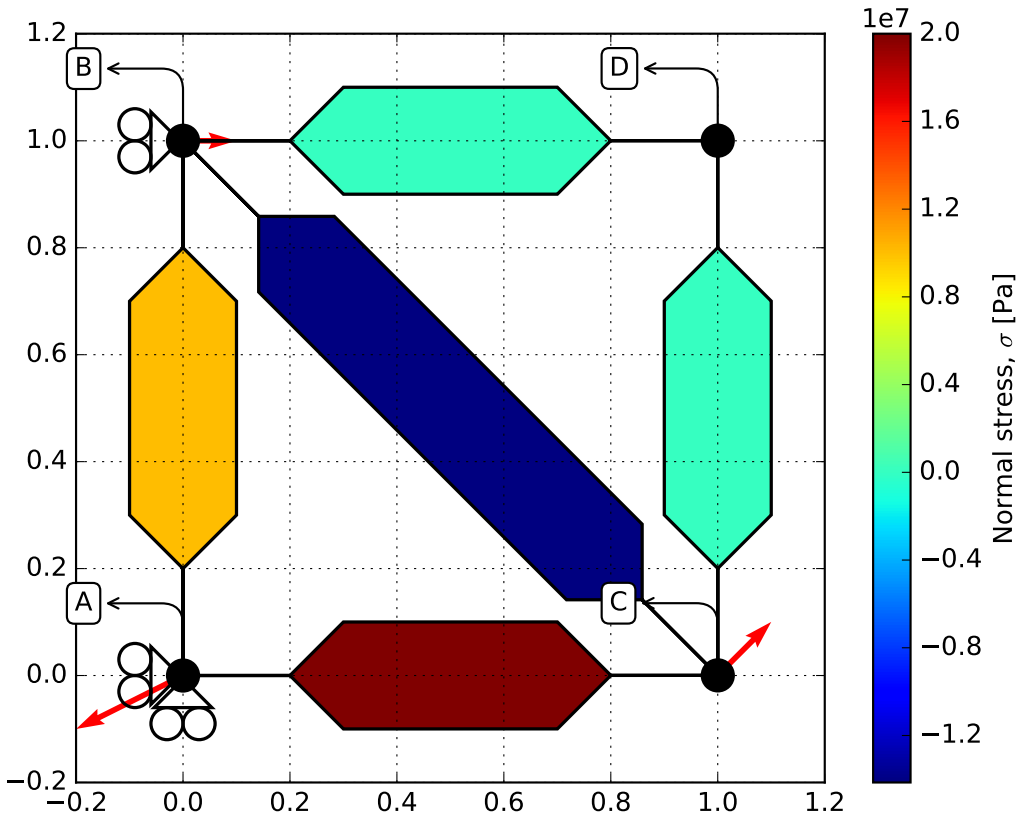
- **ax** (`matplotlib.axes instance`) – matplotlib axes.
- **deformed** (*String*) – configuration to be plotted.
- **field** – field to be plotted. Options are "tension" or "stress".
- **label** (*Bool*) – if True, node labels are plotted.
- **forces** (*Bool*) – if True, external forces are plotted.
- **displacements** (*Bool*) – if True, displacements are plotted.
- **force_scale** (*float*) – scale of the external forces vector.
- **displacement_scale** (*float*) – scale of the external displacement vector.

CHAPTER 2

Tutorial

```
import truss
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib import patches

# CONSTANTS
modulus = 210.e9      #Pa
rho = 2700.           #kg/m**3
surface = .0001        #m**2
yield_stress = 400.e6    #Pa
from scipy import optimize

# MODEL
m = truss.core.Model()

# NODES
A = m.add_node((0.,0.), label = "A")
B = m.add_node((0.,1.), label = "B")
C = m.add_node((1.,0.), label = "C")
D = m.add_node((1.,1.), label = "D")

# BOUNDARY CONDITIONS
A.block[0] = True
A.block[1] = True
B.block[0] = True

#EXTERNAL FORCES
C.force = np.array([1., 1.])*1.e3

#BARS
AB = m.add_bar(A, B, modulus = modulus, density = rho, section = surface)
AC = m.add_bar(A, C, modulus = modulus, density = rho, section = surface)
BC = m.add_bar(B, C, modulus = modulus, density = rho, section = surface)
CD = m.add_bar(C, D, modulus = modulus, density = rho, section = surface)
```

```
BD = m.add_bar(B, D, modulus = modulus, density = rho, section = surface)

#SOLVING
m.solve()

#PLOTTING
xlim, ylim = m.bbox(deformed = True)
fig = plt.figure(0)
plt.clf()
ax = fig.add_subplot(1,1,1)
ax.set_aspect("equal")
#ax.axis("off")
m.draw(ax, deformed = True, field = "stress", label = True, force_scale = 1.e-4)
plt.xlim(xlim)
plt.ylim(ylim)
plt.grid()
plt.show()
```



CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index