frame-calculator

Release 0.1.0

Oct 28, 2017

Theory

1 About finite element method **1**

2 Example 1 **3**

3 frame_calculator package **5**

3.1 Submodules 5

3.2 Module contents 9

Python Module Index **11**

CHAPTER 1

About finite element method

CHAPTER 2

Example 1

CHAPTER 3

frame_calculator package

Submodules

frame_calculator.line module

Beam element's stiffness calculations.

frame_calculator.line.**stiffness_global**(*x*, *y*, *z*, *E*, *G*, *Ax*, *Iz=0*, *Iy=0*, *Ay=0*, *Az=0*, *theta=0*, *J=0*)

Generate transformed stiffness matrixes of Timoshenko beam.

It yields stiffness matrixes converted from local axes to global axes.

Parameters

- **x** (*float*) – x coodinate of the beam directional vector.
- **y** (*float*) – y coodinate of the beam directional vector.
- **z** (*float*) – z coodinate of the beam directional vector.
- **E** (*float*) – Young's modulus (elasticity coefficients).
- **G** (*float*) – Shear modulus.
- **Ax** (*float*) – Cross-sectional area.
- **Iz** (*float*) – Moment of inertia of area around z-axis.
- **Iy** (*float*) – Moment of inertia of area around y-axis.
- **Ay** (*float*) – Cross-sectional area for y-axis shear stiffness.
- **Az** (*float*) – Cross-sectional area for z-axis shear stiffness.
- **theta** (*float*) – The rotation angle around x-axis of section (beta-angle).
- **J** (*float*) – Polar moment of area.

Yields *tuple* – 2-dimension tuples corresponding stiffness matrixes.

frame_calculator.line.**stiffness_local**(*L*, *E*, *G*, *Ax*, *Iz=0*, *Iy=0*, *Ay=0*, *Az=0*, *J=0*)
Generate stiffness matrixes of Timoshenko beam.

(TODO) Axis explaination It yields 4 matrixes M11, M12, M21, M22

Parameters

- **L** (*float*) – Length of the beam.
- **E** (*float*) – Young's modulus (elasticity coefficients).
- **G** (*float*) – Shear modulus.
- **Ax** (*float*) – Cross-sectional area.
- **Iz** (*float*) – Moment of inertia of area around z-axis.
- **Iy** (*float*) – Moment of inertia of area around y-axis.
- **Ay** (*float*) – Cross-sectional area for y-axis shear stiffness.
- **Az** (*float*) – Cross-sectional area for z-axis shear stiffness.
- **J** (*float*) – Polar moment of area.

Yields *tuple* – 2-dimension tuples corresponding stiffness matrixes.

frame_calculator.matrix module

frame_calculator.matrix.**transformMatrix**(*x*, *y*, *z*, *theta*)

frame_calculator.model module

class frame_calculator.model.**Model**(*inputModel*, *allow_overwrite=False*)
Bases: object

effective_coodinates()

effective_count()

effective_indexof(*node_id*, *coodinate*)

line_vector(*line_id*)

frame_calculator.section module

Calculate cross-sectional coefficients.

This module provides functions to calculate cross-sectional coefficients, such as area, inertia, etc. Usually, *convert* function is just you need to convert raw parameter collections to useful coefficient collections.

Example

(TODO) calculation example.

frame_calculator.section.**convert**(*section_parameters*)
Convert raw section parameters to calculated coefficients.

Calculate structural coefficients of section. Argument dictionary will be parsed according to its shape. Following shapes and functions are acceptable and used to calculate return values.

- 'H' : *frame_calculator.section.h()*
- 'I' : frame_calculator.section.i()
- 'O' : *frame_calculator.section.o()*

Parameters **section_parameters** (*dict*) – Section's shape and size. Must have 'shape' key and parameters corresponding the shape.

Returns Section's coefficients.

Return type dict

frame_calculator.section.**h**(*H*, *B*, *tw*, *tf*, *r=0*)

Calculate cross-sectional coefficients of H section.

H section is commonly used as horizontal steel beams.

Parameters

- **H** (*float*) – Height, or distance between the outedge of flanges.
- **B** (*float*) – Breadth or width of the flanges.
- **tw** (*float*) – Thickness of the web.
- **tf** (*float*) – Thickness of the flanges. 2 flanges are assumed to have same thickness.
- **r** (*float, optional*) – Radius of fillets. 4 fillets are assumed to have same radius.

Returns

Dictionary having calculated cross-sectional coefficients.:

```
{
    'Ax': Cross-sectional area,
    'Ay': Cross-sectional area of the web,
    'Az': Cross-sectional area of flanges (total),
    'Iy': Moment of inertia around y-axis,
    'Iz': Moment of inertia around z-axis,
    'J': Polor moment of area,
    'Zy': Elastic section modulus around y-axis,
    'Zz': Elastic section modulus around z-axis,
    'iy': Radius of gyration around y-axis,
    'iz': Radius of gyration around z-axis
}
```

In this context,

- x-axis direction is axial direction.
- y-axis direction is weak-axis (parallels flanges) direction.
- z-axis direction is strong-axis (parallels web) direction.

Return type dict(str, float)

frame_calculator.section.**o**(*D*, *t=0*)

Calculate cross-sectional coefficients of circle and pipe section.

The outedge of section is assumed to be a perfect circle.

Parameters

- **D** (*float*) – Diameter.

- **t** (*float, optional*) – Thickness. If t = 0, the section is considered as filled circle. Othewise as hollow.

Returns

Dictionary having calculated cross-sectional coefficients.:

```
{
    'Ax': Cross-sectional area,
    'Iy': Moment of inertia around y-axis,
    'Iz': Moment of inertia around z-axis,
    'Zy': Elastic section modulus around y-axis,
    'Zz': Elastic section modulus around z-axis,
    'iy': Radius of gyration around y-axis,
    'iz': Radius of gyration around z-axis,
    'J': Polor moment of area
}
```

In this context, x-axis direction is axial direction.

Return type dict(str, float)

frame_calculator.section.**properties**(*shape*, ***kwargs*)

Proxy to convert.

Shape and other parameters are mixed in one dictionary.

Parameters

- **shape** (*str*) – String represents section's shape.
- ****kwargs** – Parameters.

Returns Section's coefficients.

Return type dict

frame_calculator.section.**t**(*H*, *B*, *tw*, *tf*, *r=0*)

Calculate cross-sectional coefficients of T section.

T section is like cut off H section.

Parameters

- **H** (*float*) – Height, or distance between outedge of the flange and the tip of the web.
- **B** (*float*) – Breadth or width of the flange.
- **tw** (*float*) – Thickness of the web.
- **tf** (*float*) – Thickness of the flange.
- **r** (*float, optional*) – Radius of fillets. 2 fillets are assumed to have same radius.

Returns

Dictionary having calculated cross-sectional coefficients.:

```
{
    'Ax': Cross-sectional area,
    'Ay': Cross-sectional area of the web,
    'Az': Cross-sectional area of the flange,
    'Iy': Moment of inertia around y-axis,
    'Iz': Moment of inertia around z-axis,
    'Zy': Elastic section modulus around y-axis,
    'Zz': Elastic section modulus around z-axis,
```

```
    'iy': Radius of gyration around y-axis,
    'iz': Radius of gyration around z-axis,
    'Cz': Distance between the outedge of the flange and the centroid.
}
```

In this context,

- x-axis direction is axial direction.
- y-axis direction is weak-axis (parallels flanges) direction.
- z-axis direction is strong-axis (parallels web) direction.

Return type dict(str, float)

Module contents

Calculate structural efficients of frame structures.

This module provides functions to calculate behaivers of frame structure.

Example

(TODO) example usage.

frame_calculator.**calculate**(*model*)

Calculate displacements of nodes in frame structure.

[TODO] detailed discription.

- 'nodes' : { id(hashable): { x:Real, y:Real, z:Real } }
- 'lines' : { id(hashable): { n1:id, n2:id, EA:Real } }
- 'boundaries' : { id(hashable): { node:id, x:Real or Bool, y:Real or Bool, z:Real or Bool, rx:Real or Bool, ry:Real or Bool, rz:Real or Bool } }
- 'nodeLoads' : { id(hashable): { node:id, x:Real, y:Real, z:Real, rx:Real, ry:Real, rz:Real } }

Parameters **model** (*dict*) – Dictionary contains structure data.

Returns Contains displacements of nodes.

Return type dict

frame_calculator.**calculated_material**(*material_obj*, *arg_names*)

frame_calculator.**calculated_materials**(*materials*, *arg_names*)

frame_calculator.**calculated_section**(*section_obj*, *arg_names*)

frame_calculator.**calculated_sections**(*sections*, *arg_names*)

frame_calculator.**fixed_coos**(*boundary_objs*)

frame_calculator.**fixed_coos_of_boundary**(*boundary_obj*)

frame_calculator.**get_indexes**(*node_id*, *coo_indexes*)

frame_calculator.**index_dict**(*seq*)

frame_calculator.**items**(*seq*)

```
>>> item = items(["x", "y", 3])
>>> next(item)
(0, 'x')
>>> next(item)
(1, 'y')
>>> next(item)
(2, 3)
```

```
>>> item = items(("a", 4, "f"))
>>> next(item)
(0, 'a')
>>> next(item)
(1, 4)
>>> next(item)
(2, 'f')
```

frame_calculator.**keys**(*seq*)

frame_calculator.**line_node_ids**(*line_obj*)

frame_calculator.**line_nodes**(*line_obj*, *nodes*)

frame_calculator.**line_vector**(*n1_obj*, *n2_obj*)

frame_calculator.**node_vector**(*node_obj*)

frame_calculator.**stiffness_node_ids**(*line_obj*)

frame_calculator.**unfixed_coos**(*node_ids*, *boundary_objs*)

frame_calculator.**values**(*seq*)

Python Module Index

f

frame_calculator, 9
frame_calculator.line, 5
frame_calculator.matrix, 6
frame_calculator.model, 6
frame_calculator.section, 6

Index

C

calculate() (in module frame_calculator), 9
calculated_material() (in module frame_calculator), 9
calculated_materials() (in module frame_calculator), 9
calculated_section() (in module frame_calculator), 9
calculated_sections() (in module frame_calculator), 9
convert() (in module frame_calculator.section), 6

E

effective_coodinates() (frame_calculator.model.Model method), 6
effective_count() (frame_calculator.model.Model method), 6
effective_indexof() (frame_calculator.model.Model method), 6

F

fixed_coos() (in module frame_calculator), 9
fixed_coos_of_boundary() (in module frame_calculator), 9
frame_calculator (module), 9
frame_calculator.line (module), 5
frame_calculator.matrix (module), 6
frame_calculator.model (module), 6
frame_calculator.section (module), 6

G

get_indexes() (in module frame_calculator), 9

H

h() (in module frame_calculator.section), 7

I

index_dict() (in module frame_calculator), 9
items() (in module frame_calculator), 9

K

keys() (in module frame_calculator), 10

L

line_node_ids() (in module frame_calculator), 10
line_nodes() (in module frame_calculator), 10
line_vector() (frame_calculator.model.Model method), 6
line_vector() (in module frame_calculator), 10

M

Model (class in frame_calculator.model), 6

N

node_vector() (in module frame_calculator), 10

O

o() (in module frame_calculator.section), 7

P

properties() (in module frame_calculator.section), 8

S

stiffness_global() (in module frame_calculator.line), 5
stiffness_local() (in module frame_calculator.line), 5
stiffness_node_ids() (in module frame_calculator), 10

T

t() (in module frame_calculator.section), 8
transformMatrix() (in module frame_calculator.matrix), 6

U

unfixed_coos() (in module frame_calculator), 10

V

values() (in module frame_calculator), 10