2DFrame – Python

Hướng dẫn sử dụng

Phạm Hoàng Anh
Bộ môn Cơ học kết cấu
Trường Đại học Xây dựng Hà Nội
Email: *anhph2@huce.edu.vn*

2DFrame là bộ chương trình máy tính để tính toán kết cấu hệ thanh 2D theo phương pháp phần tử hữu hạn (FEM). Chương trình có thể được sử dụng cho các bài toán phân tích kết cấu dàn phẳng và khung phẳng, với mục đích chính là phục vụ giảng dạy và nghiên cứu.

2DFrame được phát triển bởi TS. Phạm Hoàng Anh, Bộ môn Cơ học kết cấu, Trường Đại học Xây dựng Hà Nội, bắt đầu từ năm 2014 cho MATLAB. Phiên bản 2DFrame – Python được phát triển năm 2024 dùng cho Python. Khác với phiên bản cho MATLAB (được phát triển hoàn toàn độc lập), phiên bản Python của 2DFrame được phát triển trên nền bộ chương trình CALFEM của trường Đại học Lund.

1. Cài đặt

Yêu cầu:

Để có thể sử dụng 2DFrame - Python, bạn cần có môi trường Python cài trên máy tính và thư viện CALFEM-Python.

Cài đặt 2DFrame

- Cài đặt CALFEM: `pip install calfem-python`
- Cài đặt 2DFrame: copy và giải nén file `2DFrame-Python.zip` vào một thư mục trên máy tính. Vào môi trường Python để đặt đường dẫn đến thư mục chứa file `Module_2DFrame.py`.

2. Bắt đầu với 2DFrame

Để sử dụng 2DFrame, bạn cần nạp mô đun này vào mỗi chương trình tính toán. Sử dụng câu lệnh:

```
import Module_2DFrame as st
```

Mô hình kết cấu

Mô hình kết cấu được khởi tạo bằng lệnh `st.SystemModel()`. Tất cả các đặc trưng của mô hình, như các nút, phần tử, vật liệu và lực… đều được lưu giữ trong mô hình.

Cú pháp:

```
model = st.SystemModel(str_type)
```

Biến `str_type` là loại kết cấu. Có 3 loại kết cấu mà 2DFrame hỗ trợ là:

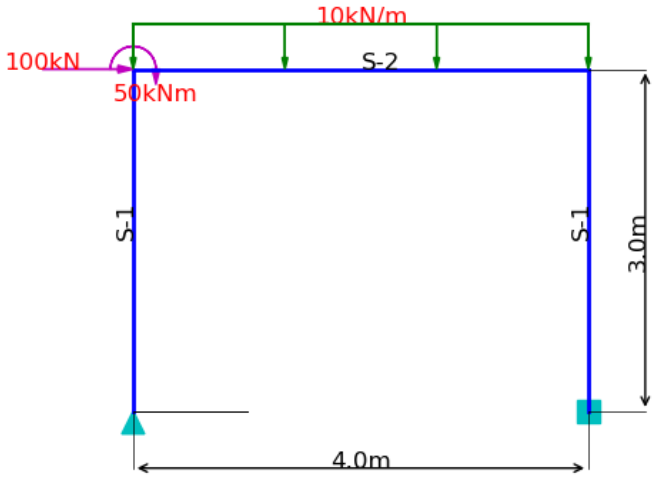
- Khung 2D: `str_type='2Dframe'`
- Dàn 2D: `str_type='2Dtruss'`
- Dàn 3D: `str_type='3Dtruss'`

Các cấu trúc dữ liệu trong mô hình `SystemModel`:

`type`	- loại kết cấu (giá trị mặc định là `'2Dframe'`)
`ndof`	- số thành phần chuyển vị (bậc tự do) của một nút, mặc định là 3
`nsec`	- số tiết diện tính nội lực và chuyển vị cho phần tử, mặc định là 11
`Node`	- bảng dữ liệu tọa độ nút
`Ele`	- bảng dữ liệu phần tử
`Mat`	- bảng dữ liệu tiết diện
`Bound`	- bảng dữ liệu liên kết tựa
`Spring`	- Bảng dữ liệu gối đàn hồi
`Eload`	- bảng dữ liệu tải trọng tác dụng trên phần tử
`Nload`	- bảng dữ liệu tải trọng tác dụng tại nút
`Supp`	- bảng dữ liệu về chuyển vị gối tựa
`disp`	- kết quả tính chuyển vị nút
`force`	- kết quả tính nội lực trong phần tử
`res`	- kết quả phản lực liên kết tựa

Ví dụ minh họa:

Thực hiện phân tích hệ kết cấu khung 2D chịu tải trọng cho trên Hình 1.



S-1: A1 = 25.0e-4; I1 = 1.0e-5

S-2: A2 = 25.0e-4; I2 = 1.0e-5

E = 2.1e+11

Hình 1. Sơ đồ khung chịu tải trọng

Đầu tiên, ta cần khởi tạo mô hình kết cấu khung phẳng 2D:

```
model = st.SystemModel('2Dframe')
```

Sau đó, ta nhập các số liệu cho mô hình kết cấu:

- Nhập bảng dữ liệu tọa độ nút:

```
model['Node'] = [[0, 0],
                 [4, 0],
                 [0, 3],
                 [4, 3]]
```

- Nhập bảng dữ liệu phần tử:

```
model['Ele'] = [[1, 3, 1, 1, 1],
                [2, 4, 1, 1, 1],
                [3, 4, 2, 1, 1]]
```

- Nhập thông số vật liệu và tiết diện:

```
E = 2.1e11;
A1 = 25.0e-4; I1 = 1.0e-5
A2 = 25.0e-4; I2 = 1.0e-5
model['Mat'] = [[E, A1, I1],
                [E, A2, I2]]
```

- Nhập bảng dữ liệu liên kết:

```
model['Bound'] = [[1, 1, 1, 0],
                  [2, 1, 1, 1]]
```

- Nhập bảng tải trọng trên phần tử:

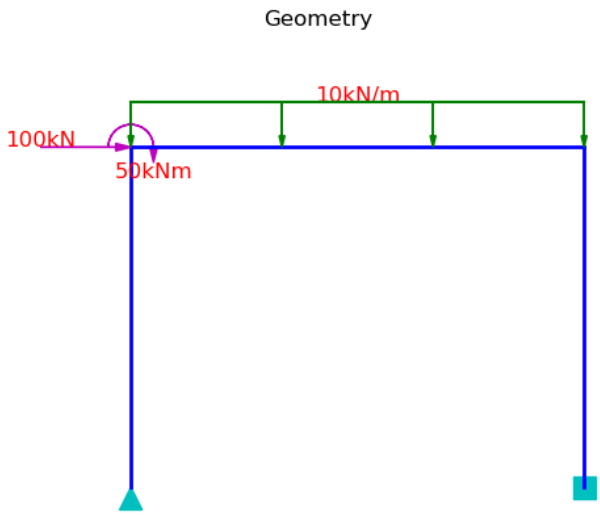
```
model['Eload'] = [[3, 0, -10]]
```

- Nhập bảng tải trọng tại nút:

```
model['Nload'] = [[3, 100, 0, -50]]
```

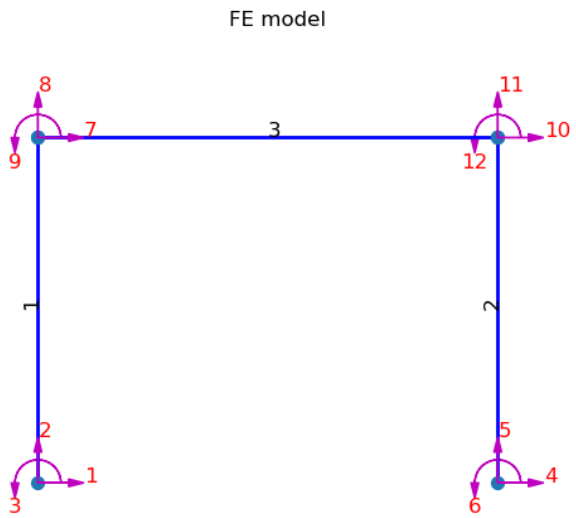
Đến đây, ta đã hoàn thành nhập số liệu vào mô hình. Để xem mô hình kết cấu, sử dụng câu lệnh:

```
st.show_geometry(model)
```



Ta cũng có thể xem sơ đồ rời rạc (sơ đồ PTHH) bằng lệnh:

```
st.show_FEM(model)
```



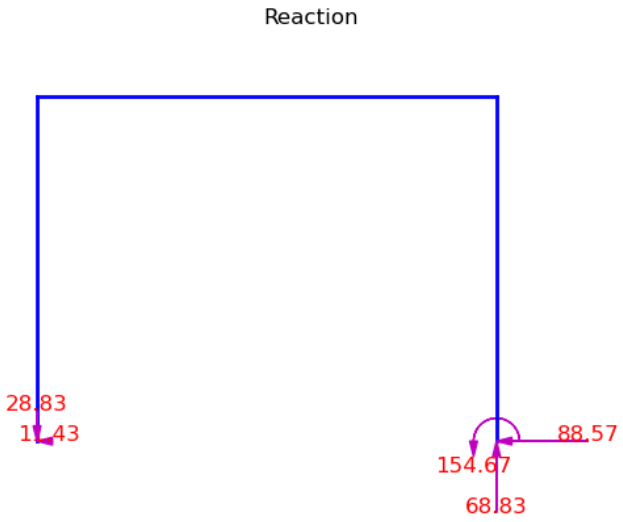
Tiếp theo, thực hiện phân tích kết cấu:

```
model = st.Solve2Dframe(model)
```

Bây giờ, ta có thể biểu diễn các kết quả tính toán được:

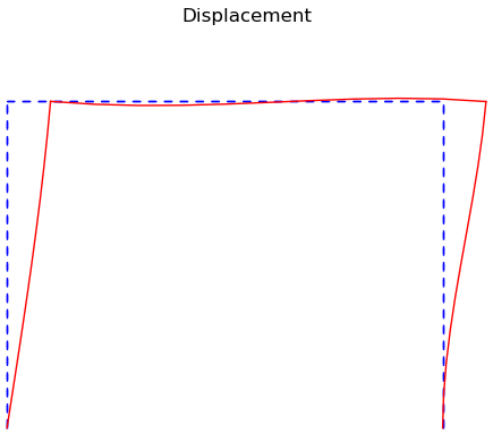
- Phản lực tại liên kết tựa:

```
st.show_reaction(model)
```



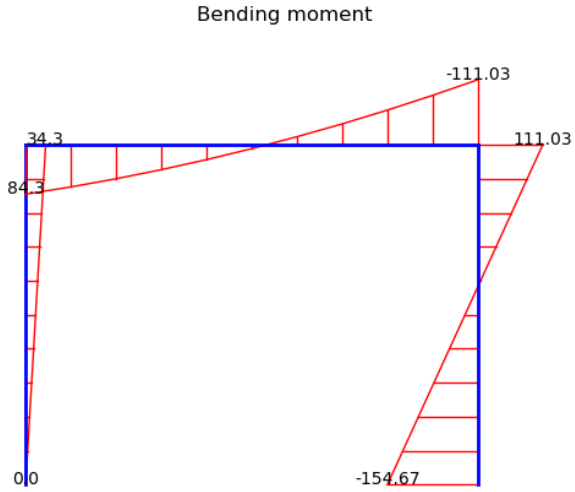
- Sơ đồ biến dạng:

```
st.show_displacement(model)
```



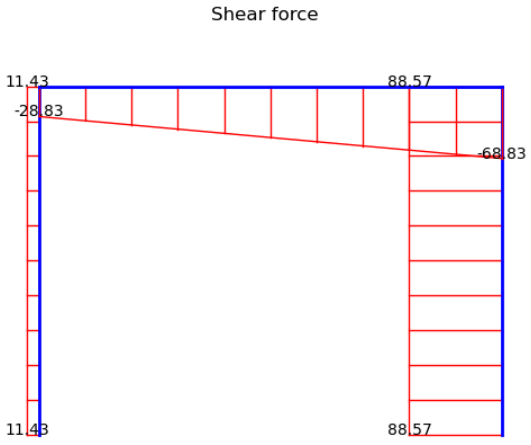
- Biểu đồ mô men uốn:

```
st.show_moment(model)
```



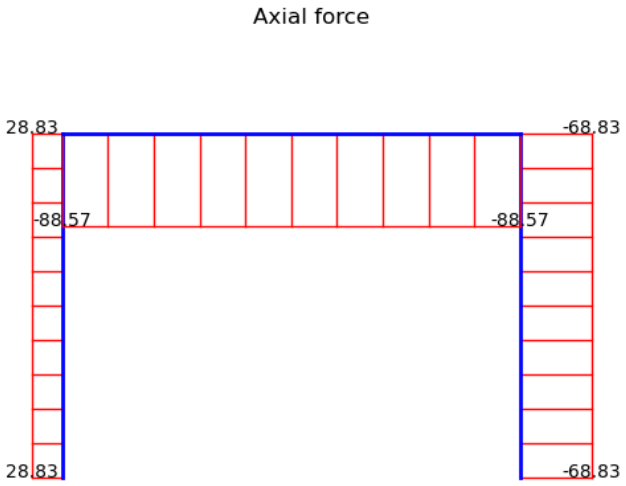
- Biểu đồ lực cắt:

```
st.show_shear(model)
```



- Biểu đồ mô lực dọc:

```
st.show_axial(model)
```



3. Nút

Dữ liệu về nút kết cấu được nhập và lưu trong biến `Node` của mô hình dưới dạng bảng có kích thước *NoNx2*, trong đó *NoN* là số lượng nút. Mỗi hàng của `Node` chứa 2 tham số tương ứng với các tọa độ của nút theo 2 phương x, y. Tên (số thứ tự) nút được mặc định là số thứ tự của hàng.

Khai báo nút

Khai báo tất cả các nút của hệ dưới dạng bảng, ví dụ khai báo 4 nút của ví dụ trên như sau:

```
model['Node'] = [[0, 0],
                 [4, 0],
                 [0, 3],
                 [4, 3]]
```

4. Phần tử

Dữ liệu phần tử kết cấu được lưu trong biến `Ele` của mô hình dưới dạng bảng có kích thước *NoEx5*, trong đó *NoE* là số lượng phần tử. Cấu trúc và các tham số của phần tử như sau:

`[in, jn, m, si, sj]`

`in` - tên nút đầu i

`jn` - tên nút đầu j

`m` - tên (số thứ tự) loại tiết diện

`si` - mã liên kết đầu i của thanh (0 tương ứng với khớp, 1 tương ứng với hàn)

`sj` - mã liên kết đầu j của thanh (ý nghĩa giống `si`)

Khai báo phần tử

Khai báo tất cả các phần tử của hệ dưới dạng bảng, ví dụ khai báo cho 3 phần tử ở ví dụ trên:

```
model['Ele'] = [[1, 3, 1, 1, 1],
                [2, 4, 1, 1, 1],
                [3, 4, 2, 1, 1]]
```

Chú ý: Trường hợp hệ dàn, mã liên kết đầu thanh không cần đặt là 0 nữa.

5. Vật liệu và tiết diện

Dữ liệu về tiết diện được lưu trong biến `Mat` của mô hình dưới dạng bảng có kích thước *NoMx3*, trong đó *NoM* là số loại tiết diện. Các tham số tiết diện được cấu trúc như sau:

```
[E, A, I]
E    - mô đun đàn hồi
A    - diện tích tiết diện
I    - mô men quán tính tiết diện
```

Ví dụ khai báo 2 loại tiết diện của khung trong ví dụ trên theo cấu trúc bảng như sau:

```
model['Mat'] = [[E, A1, I1],
                [E, A2, I2]]
```

Câu lệnh trên sẽ nhập vào mô hình 2 loại tiết diện. Thứ tự của tiết diện khai báo trong Mat sẽ được dùng để khai báo dữ liệu cho biến m của Ele

6. Liên kết tựa

Bảng dữ liệu `Bound` trong mô hình được dùng để khai báo và lưu thông tin về các liên kết tựa của kết cấu. Cấu trúc và các tham số như sau:

```
[node, c1, c2, c3]
node - tên nút
c1   - điều kiện liên kết theo phương x
c2   - điều kiện liên kết theo phương y
c3   - điều kiện liên kết mô men (khung 2D)
```

Các tham số c1, c2, c3 sẽ lấy giá trị bằng 1 nếu có liên kết, và bằng 0 nếu không có liên kết.

Ví dụ:

- Gối di động thẳng đứng: `c1=0; c2=1; c3=0;`
- Gối cố định: `c1=1; c2=1; c3=0;`
- Ngàm: `c1=1; c2=1; c3=1;`

Câu lệnh sau sẽ khai báo liên kết gối cố định ở nút 1 và ngàm ở nút 2 của khung trong ví dụ bên trên:

```
model['Bound'] = [[1, 1, 1, 0],
                  [2, 1, 1, 1]]
```

7. Tải trọng

Lực tác dụng tại nút

Lực tác dụng tại nút được nhập vào bảng dữ liệu `Nload` của mô hình với cấu trúc và các tham số như sau:

```
[node, Fx, Fx, Mz]
node - tên nút
Fx   - lực tập trung theo phương x
Fy   - lực tập trung theo phương y
Mz   - mô men tập trung (khung 2D)
```

Ví dụ, để khai báo lực tập trung 100 kN theo phương ngang và mô men tập trung -50 kNm tác dụng tại nút 3, ta nhập câu lệnh:

```
model['Nload'] = [[3, 100, 0, -50]]
```

Lực tác dụng trên phần tử

Lực tác dụng trên phần tử được nhập vào bảng dữ liệu `Eload` trong mô hình với cấu trúc và các tham số như sau:

```
[ele, qi, qj]
```

`ele` - tên phần tử

`qi` - lực phân bố đều theo phương dọc trục thanh

`qj` - lực phân bố đều theo phương vuông góc với trục thanh

Ví dụ: với kết cấu ở ví dụ bên trên, thanh 3 chịu lực phân bố đều 10 kN/m theo phương thẳng đứng hướng xuống dưới, câu lệnh nhập tải trọng vào mô hình như sau:

```
model['Eload'] = [[3, 0, -10]]
```

8. Chuyển vị liên kết và gối đàn hồi

Chuyển vị liên kết tựa

2DFrame cho phép tính toán kết cấu khi có chuyển vị cưỡng bức tại liên kết tựa nếu ta khai báo bảng dữ liệu `Supp` vào mô hình tính. Lưu ý, chỉ các liên kết tựa đã được khai báo ở `Bound` mới có thể khai báo chuyển vị cưỡng bức. Cấu trúc nhập liệu tương tự như khai báo các liên kết tựa như sau:

```
[node, Zx, Zy, Zz]
```

`node` - tên nút

`Zx` - chuyển vị theo phương x

`Zy` - chuyển vị theo phương y

`Zz` - chuyển vị xoay (khung 2D)

Ví dụ, để khai báo chuyển vị cưỡng bức bằng 0.01m theo phương thẳng đứng lại nút 1 ta nhập câu lệnh:

```
model['Supp'] = [[1, 0, -0.01, 0]]
```

Gối tựa đàn hồi

Để khai báo gối đàn hồi, ta nhập bảng dữ liệu `Spring` vào mô hình. Cấu trúc nhập liệu tương tự như khai báo các liên kết tựa như sau:

```
[node, kx, ky, kz]
```

`node` - tên nút

`kx` - độ cứng gối đàn hồi phương x

`ky` - độ cứng gối đàn hồi phương y

`kz` - độ cứng gối đàn hồi xoay (khung 2D)

Ví dụ:

```
model['Spring'] = [[1, 0, 0, 1000]]
```

9. Biểu diễn kết quả dạng hình vẽ

2DFrame cung cấp các kết quả dưới dạng hình vẽ cho nội lực và các phản lực liên kết tựa.

- Sơ đồ kết cấu: `show_geometry(model)`
- Sơ đồ rời rạc: `show_FEM(model)`
- Phản lực liên kết: `show_reaction(model)`
- Sơ đồ biến dạng: `show_displacement(model, scale)`
- Biểu đồ mô men uốn: `show_moment(model, scale)`
- Biểu đồ lực cắt: `show_shear(model, scale)`
- Biểu đồ lực dọc: `show_axial(model, scale)`

Tham số `scale` là tỉ lệ vẽ. Trường hợp đơn giản nhất, ta gán giá trị `scale` là `'Auto'`, chương trình sẽ tự động tính toán tỷ lệ vẽ phù hợp với kích thước hình học của kết cấu.

10. Trích xuất số liệu và kết quả

Để trích xuất các số liệu của mô hình cũng như kết quả tính toán, ta truy cập vào các bảng dữ liệu tương ứng của mô hình theo cú pháp: `model['name']`, trong đó `model` là đối tượng mô hình, `name` là tên bảng dữ liệu. Ví dụ, để hiện giá trị chuyển vị tại các nút, ta nhập câu lệnh:

```
model['disp']
```

Ngoài ra, 2DFrame cung cấp một số câu lệnh để biểu diễn các kết quả tính toán:

- Chuyển vị nút: `disp_ndisp(model,list)`

Trong đó `list` là danh sách các nút cần biểu diễn chuyển vị (giá trị mặc định là `list=None` sẽ biểu diễn chuyển vị của tất cả các nút). Ở ví dụ bên trên, để biểu diễn chuyển vị của nút 3 và 4, ta thực hiện lệnh:

```
st.disp_ndisp(model,[3,4])
```

```
Node 3
+------------+------------+-------------+
|          dx |          dy |           fi |
|------------+------------+-------------|
| 1.4233e-04 | 1.6475e-07 | -3.1112e-05 |
+------------+------------+-------------+
Node 4
+------------+-------------+-------------+
|          dx |           dy |           fi |
|------------+-------------+-------------|
| 1.4166e-04 | -3.9332e-07 | -3.1175e-05 |
+------------+-------------+-------------+
```

- Nội lực đầu phần tử: `disp_eforce(model,list)`

Trong đó `list` là danh sách các phần tử (giá trị mặc định là `list=None` sẽ biểu diễn nội lực tại đầu của tất cả các phần tử). Ví dụ:

```
st.disp_eforce(model)
```

Element 1

N	Q	M
2.8831e+01	1.1432e+01	0.0000e+00
2.8831e+01	1.1432e+01	3.4297e+01

Element 2

N	Q	M
-6.8831e+01	8.8568e+01	-1.5467e+02
-6.8831e+01	8.8568e+01	1.1103e+02

Element 3

N	Q	M
-8.8568e+01	-2.8831e+01	8.4297e+01
-8.8568e+01	-6.8831e+01	-1.1103e+02

11. Một số tùy chọn