重庆大学
CHONGQING UNIVERSITY

基于ABAQUS的参数化建模

建筑工业化与智能化课题组

汇报人：王禄锋

2021.12

目 录

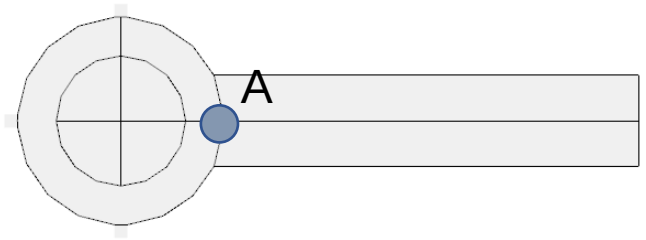
一．ABAQUS相关基础知识

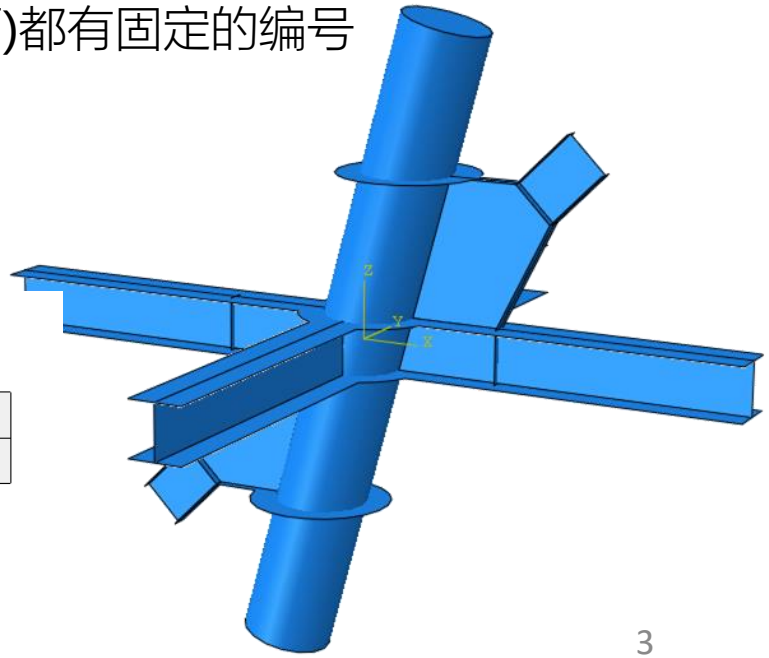
1. 建模命令流文件 `abaqus.rpy` 直接改变其文件后缀为 `.py` 可成为Python文件

2. 在python文件中，可将其中的梁长1000mm替换为 x，从而构造出以梁长 x 为参数的函数 $f(x)$进行建模

3. 具有开放的接口，可以在Windows终端通过 `abaqus cae noGUI=script.py` 等命令进行调用

4. ABAQUS在模型Assembly之后形成几何交点(面)，所有的交点(面)都有固定的编号
   在没有缺少或者没有新的交点(面)出现时，其 **编号没有变化**
   是ABAQUS能够参数化建模的关键，被用来设置参考面，参考点





二. Python相关基础知识

1. 函数def

是带有名字的代码块，用于完成具体的工作。
方便多次调用。

```
def  say_hi(person):
    if person is {a old friend}:
        say('Hello,Long time no see')
     else:
        say('Hi, Nice to meet you')


say_hi('Liao yue')
>>>Hello,Long time no see
say_hi('SADJIS[H]')
>>>Hi, Nice to meet you
...
```

2. 类Class

多个函数有相同的变量。

```
class a_boy_meet():

    def __init__(self, person):
        self.person = person

    def say_hi(self):
        if self.person is {a old friend}:
        say('Hello,Long time no see')
     else:
        say('Hi, Nice to meet you')

    def make_action(self):
        if self.person is handsome:
            Leave right now!
        elif self.person is beautiful:
            try to break the Ice
        else:
            to be a true friend

LiaoYue = a_boy_meet('ZhouCao')
LiaoYue.say_hi()
LiaoYue.make_action()

>>>
'Hello,Long time no see'
 Leave right now!
```

二. Python相关基础知识

3. ABAQUS中的字典（Repository容器）

```
dictionary_information = {'Zhang Ming':[18,170,140,135,120],
                          'Zhao Yang':'like soprt'}

dictionary_information['Zhang Ming']
>>>[18,170,140,135,120]
```

通过（键key）而不是（位置index）来访问（元素value）
时间复杂度为 $O(1)$

```
regionPost = odb.rootAssembly.elementSets[setName]
```

Repository容器负责同一类型的对象的存储，可以视为python中的字典

二. Python相关基础知识

4. ABAQUS中的**函数**和**类**以及**字典**

```
import sketch
import part
import assembly
import material
import visualization
```

```
mdb.models[crash].Material[steel]
mdb.models[crash].materials[steel].Elastic(table=((30000000.0, 0.3), ))
elasticityType = mdb.models[crash].materials[steel].elastic.type
```

当它首字母大写（通常作为单数）出现时一般是构造**函数（方法）**，
而当它全小写（通常作为复数）出现时，它一般是类似**字典**的容器**（属性）**。

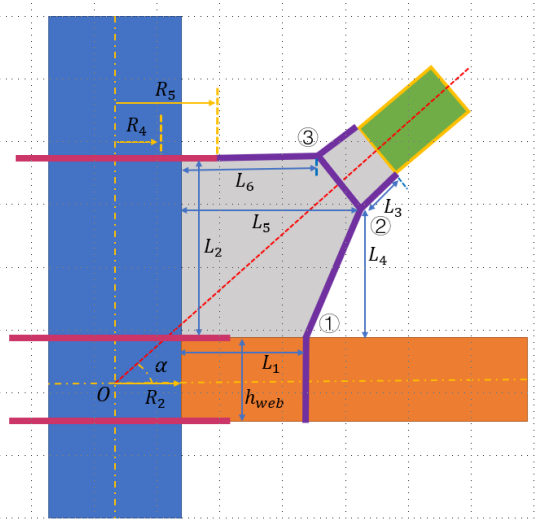
三. 参数化建模流程

前处理

- 材料选择
- 材料本构
- 几何尺寸
- ...
- 接触方式

输入参数

固定参数



使用Python2语言进行ABAQUS建模

1. 编写所需要的函数（材料的规范本构，计算直线与圆交点等）
2. 处理几何关系，计算坐标位置，以及可能用到的旋转轴的坐标以及方向。
3. 使用ABAQUS界面进行操作，在命令流文件中进行标注，建立函数进行参数化，进行测试。

后处理

1. 根据规范编写数据处理程序
2. 利用odb文件读取所需要的原始数据。
3. 对原始数据进行保存
4. 对原始数据进行处理并绘图
5. 保存计算结果

四.以倾斜钢管混凝土柱支托式斜撑为例

4.2 将需要的规范写入程序方便调用

《混凝土结构设计规范》GB 50010-2010

```
constitution.py

def get_steel_yield(steel):...

def get_steel_strength(steel):...

def get_concrete_parameters(concrete):...

def get_concrete_tensile_constitution(Ec, Ftr):...
```

```
from constitution import *
```

四.以倾斜钢管混凝土柱支托式斜撑为例

4.1 相关规范和论文确定模型的参数

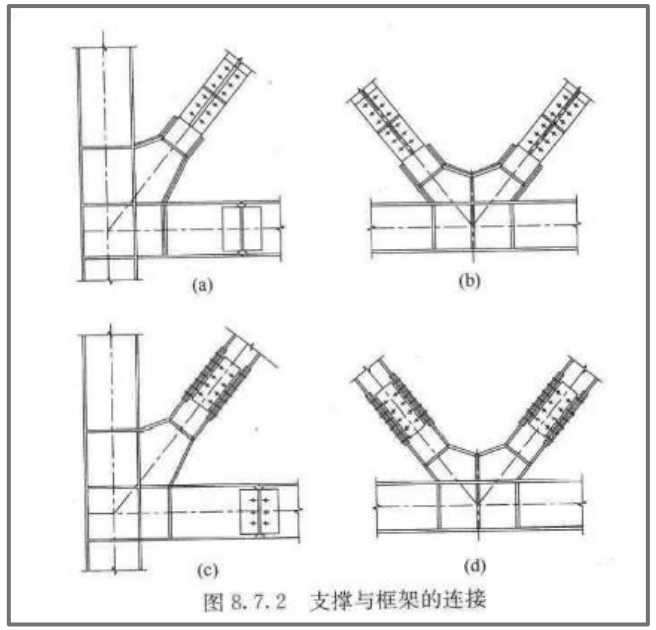


图 8.7.2 支撑与框架的连接

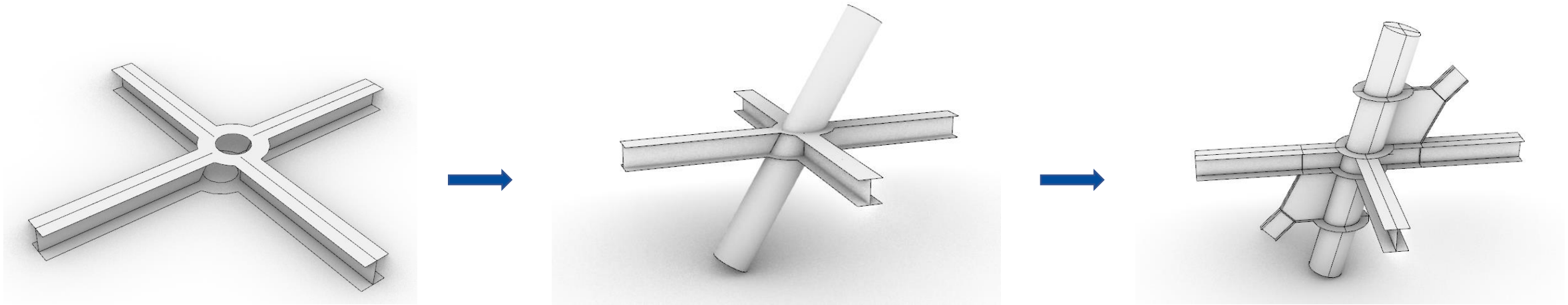
《JGJ99-2015 高层民用建筑钢结构技术规程》

项目	参数
钢材本构	双折线
混凝土本构	塑性损伤
膨胀角	40°
偏心率	0.1
形状系数	0.6667
…	…
断裂能	$G_f = \alpha(0.1f_C)^{0.7}$

《大直径钢管混凝土柱-H型钢梁框架节点的抗震性能》

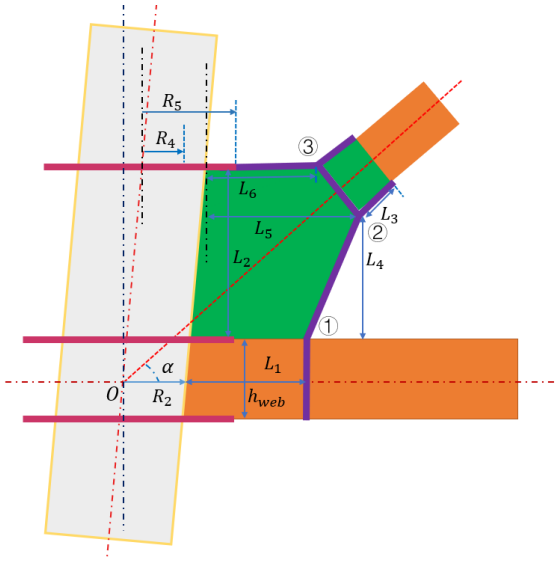
四.以倾斜钢管混凝土柱支托式斜撑为例

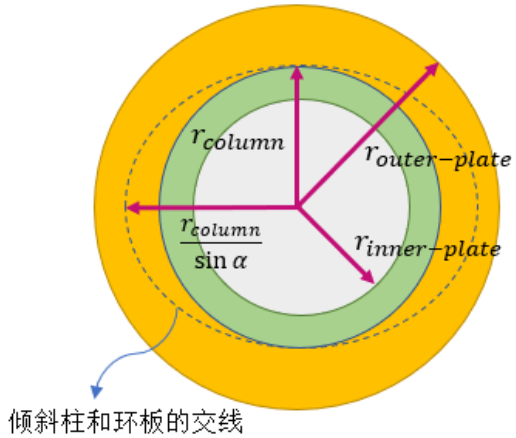
4.2 处理几何关系

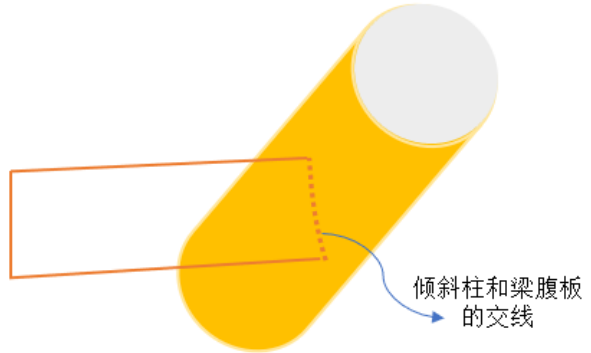


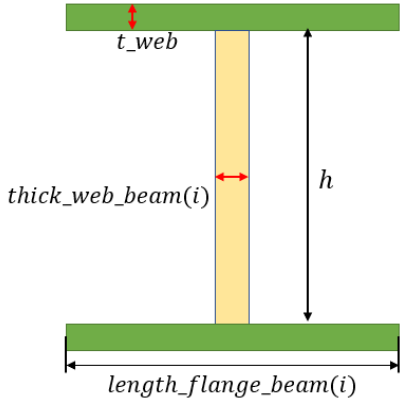
四.以倾斜钢管混凝土柱支托式斜撑为例

4.2 处理几何关系

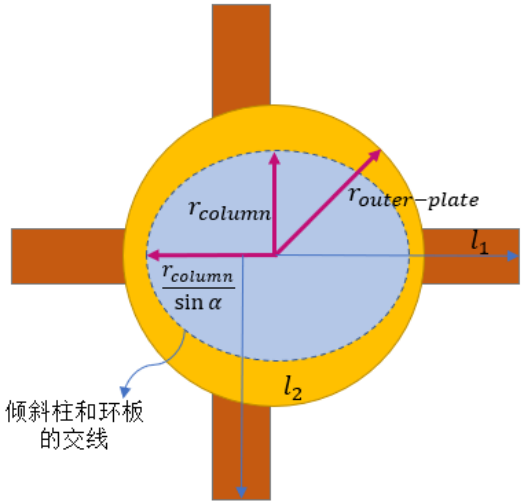


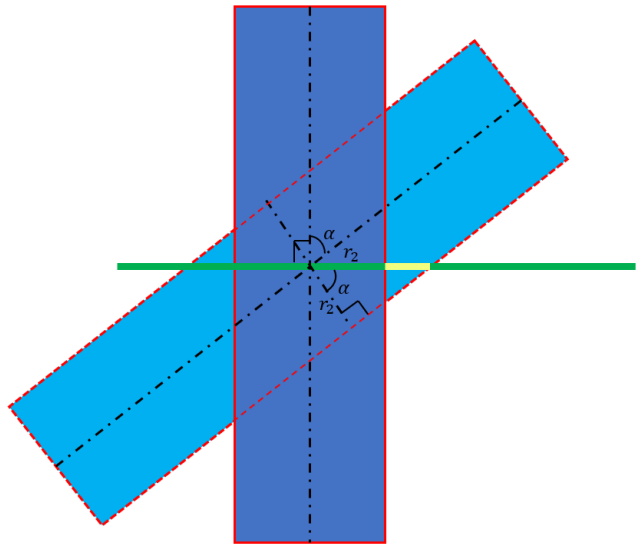






$i = 1,2,3,4$





四.以倾斜钢管混凝土柱支托式斜撑为例

4.3 添加几何处理所需要的函数

```
joint_type32_make_model.py

def get_cross_line_circle(r, h):...
    #求直线与圆的交点

def get_cross_line1_line2(k1, point1, k2, point2):...
    #已知两条直线的一点和斜率求交点

def get_cross_line_circle(point, k, r):...
    #求距离直线k上一点point距离为r的两个点

def get_ellipse(r, slope):...
    #返回椭圆的相关参数

def get_support_board_point(debut_point, r2, h_web, l1, l2, l3, l4, l5, l6, alpha,
h_deep, h_web_support):...
    #输入参数返回在ABABQUS建支撑所需要的参数
```

四.以倾斜钢管混凝土柱支托式斜撑为例

4.4 函数化建模过程

joint_type32_make_model.py

```
class joint32_model():

    def __init__(self, geometry_column, geometry_plate, num_beams,
                 geometry_beam1, geometry_beam2, geometry_beam3, geometry_beam4,
                 num_supports, geometry_support_board, support_compress,
                 material_property, material_support, mesh_size, compute_set):...

    def step1_part(self):...

    def step2_assembly(self):...

    def step3_property(self):...

    def step4_mesh(self):...

    def step5_RP(self):...

    def step6_interation(self):...

    def step7_step_out(self):...

    def step8_load(self):...

    def step9_job_submit(self):...
```

Code 1

```
def __init__(self, geometry_column, geometry_plate, num_beams,
             geometry_beam1, geometry_beam2, geometry_beam3, geometry_beam4,
             num_supports, geometry_support_board, support_compress,
             material_property, material_support, mesh_size, compute_set):

self.num_beams = num_beams
self.h_column, self.circle1, self.circle2, self.circle3,\
        self.t_tube, self.slope= geometry_column
self.h_web, self.t_plate = geometry_plate
self.l_flange1, self.w_flange1, self.t_web1 = geometry_beam1
self.l_flange2, self.w_flange2, self.t_web2 = geometry_beam2
self.l_flange3, self.w_flange3, self.t_web3 = geometry_beam3
self.l_flange4, self.w_flange4, self.t_web4 = geometry_beam4
self.mesh_size_steel, self.mesh_size_concrete = mesh_size
self.steel_plastic_tuple, self.concrete_elastic_modu,\
        self.concrete_compres_tuple = material_property

self.memory_percent, self.num_Cpus, self.num_GPUS = compute_set

self.num_supports = num_supports
self.r4, self.r5, self.debut_point, self.l1, self.l2, self.l3, self.l4, self.l5,
self.l6,self.alpha, self.t_support_board, \
        self.w_flange_support, self.t_support_flange = geometry_support_board
self.steel_strength_support, self.support_steel_plastic_tuple = material_support
self.r2 = self.circle2
self.h_web = self.h_web

self.support_compress = support_compress
```

Code 2

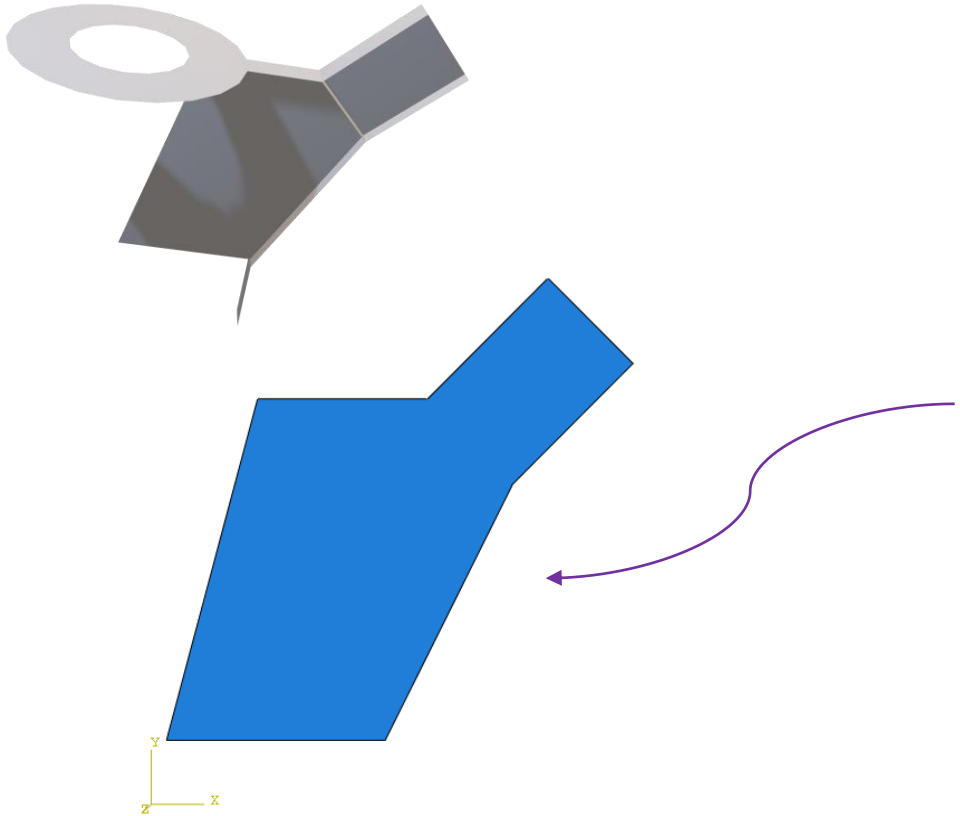
```
if __name__ == '__main__':

    geometry_support_board = [200.0, 300.0, (0.0, 0.0),
                              300.0, 800.0, 400.0, 600.0,
                              600.0, 400.0, 45, 4.0, 100.0, 4.0]
    ...

    joint_example = joint26_model(geometry_column_1, ...)
    joint_example.step1_part()
    joint_example.step2_assembly()
    joint_example.step3_property()
    joint_example.step4_mesh()
    joint_example.step5_RP()
    ...
```

Code 3

四.以倾斜钢管混凝土柱支托式斜撑为例

4.4 函数化建模过程

```
def step1_part(self):...
```



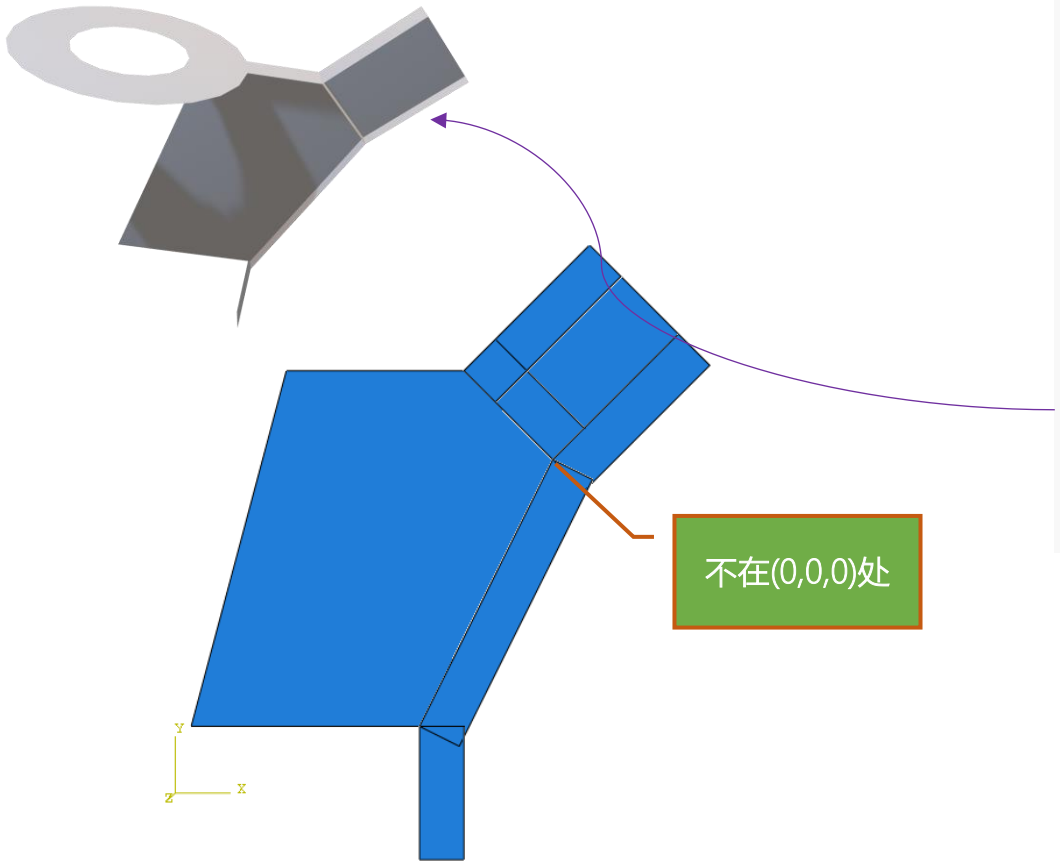
```
# support-board
cross_right_bottom, cross_right_bottom2 \
    = get_cross_line_circle((self.r2 + self.l5, self.h_web / 2 \
                             + self.l4),np.tan(self.alpha / 180.0 * np.pi), self.l3)

cross_right_up, cross_right_up2 \
           = get_cross_line_circle((self.r2 + self.l6, self.h_web / 2 + self.l2),
                                   np.tan(self.alpha / 180.0 * np.pi), self.l3)
```

```
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',sheetSize=8000.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
x_dis_board = self.l2 / np.tan((90.0 - self.slope) / 180.0 * np.pi)
s.Line(point1=(self.r2 - x_dis_board, self.h_web/2),
               point2=(self.r2 + self.l1, self.h_web/2))
s.HorizontalConstraint(entity=g[2], addUndoState=False)
s.Line(point1=(self.r2 + self.l1, self.h_web/2),
       point2=(self.r2 + self.l5, self.h_web/2 + self.l4))
s.Line(point1=(self.r2 + self.l5, self.h_web/2 + self.l4),
               point2=cross_right_bottom)
s.Line(point1=cross_right_bottom, point2=cross_right_up)
s.Line(point1=cross_right_up, point2=(self.r2 + self.l6, self.h_web / 2 \
                                      + self.l2))
s.Line(point1=(self.r2 + self.l6, self.h_web / 2 + self.l2),
               point2=(self.r2, self.h_web / 2 + self.l2))
s.HorizontalConstraint(entity=g[7], addUndoState=False)
s.Line(point1=(self.r2, self.h_web / 2 + self.l2),
               point2=(self.r2 - x_dis_board, self.h_web/2))
# s.VerticalConstraint(entity=g[8], addUndoState=False)
# s.PerpendicularConstraint(entity1=g[7], entity2=g[8], addUndoState=False)
p = mdb.models['Model-1'].Part(name='Part-support-board', dimensionality=THREE_D,
                               type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Part-support-board']
p.BaseShell(sketch=s)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Part-support-board']
del mdb.models['Model-1'].sketches['__profile__']
```

四.以倾斜钢管混凝土柱支托式斜撑为例

4.4 函数化建模过程

```
def step2_assembly(self):...
```




```
point_1 = (self.r2 + self.l1, self.h_web / 2, 0)
point_2 = (self.r2 + self.l5, self.h_web / 2 + self.l4, 0)
point_3 = (self.r2 + self.l6, self.h_web / 2 + self.l2, 0)
k_support = np.tan(self.alpha / 180.0 * np.pi)
k_support_vertical = -1 / k_support
k_2_3 = (point_3[1] - point_2[1]) / (point_3[0] - point_2[0])
k_1_2 = (point_2[1] - point_1[1]) / (point_2[0] - point_1[0])
```

```
a = mdb.models['Model-1'].rootAssembly
a.rotate(instanceList=('Part-flange-4-1',), axisPoint=point_3,
        axisDirection(1.0, k_support, 0.0), angle=90.0)
a = mdb.models['Model-1'].rootAssembly
a.translate(instanceList=('Part-flange-4-1',),
          vector=(0.0, 0.0, self.w_flange_support / 2))

a = mdb.models['Model-1'].rootAssembly
a.rotate(instanceList=('Part-flange-3-1',), axisPoint=point_2,
         axisDirection=(1.0, k_support, 0.0), angle=90.0)
a = mdb.models['Model-1'].rootAssembly
a.translate(instanceList=('Part-flange-3-1',),
            vector=(0.0, 0.0, self.w_flange_support / 2))
```

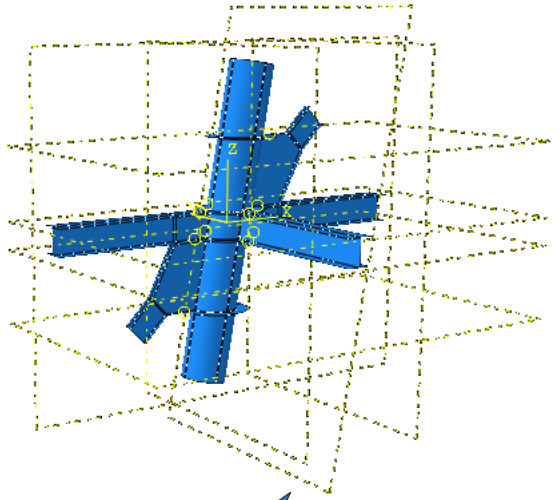
• • •

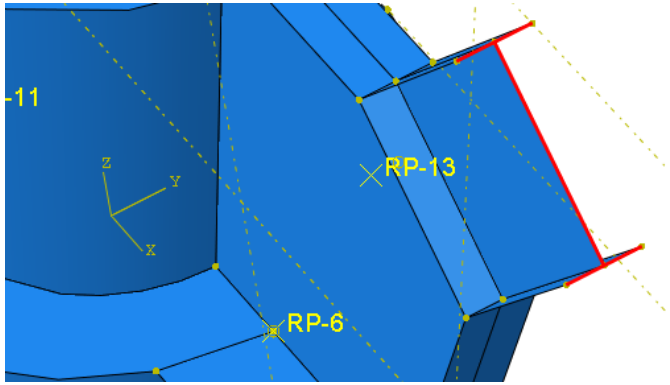
```
a = mdb.models['Model-1'].rootAssembly
a.rotate(instanceList=('Part-flange-2-1', 'Part-flange-3-1',
                       'Part-flange-4-1',
                       'Part-flange-5-1', 'Part-support-board-1'),
         axisPoint=(0.0, 0.0, 0.0),
         axisDirection=(1.0, 0.0, 0.0), angle=90.0)
```

四.以倾斜钢管混凝土柱支托式斜撑为例

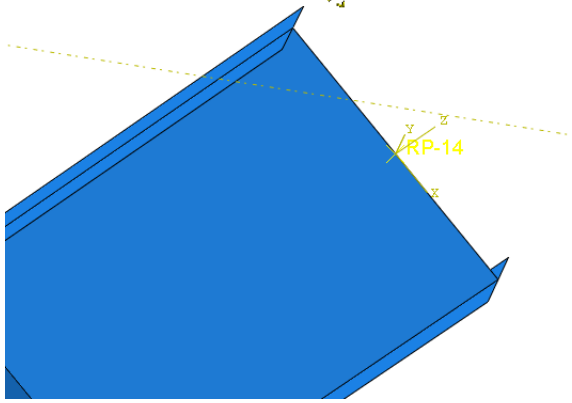
4.4 函数化建模过程

```
def step4_mesh(self):...

def step5_RP(self):...

def step6_interation(self):..

def step7_step_out(self):...

def step8_load(self):...
```

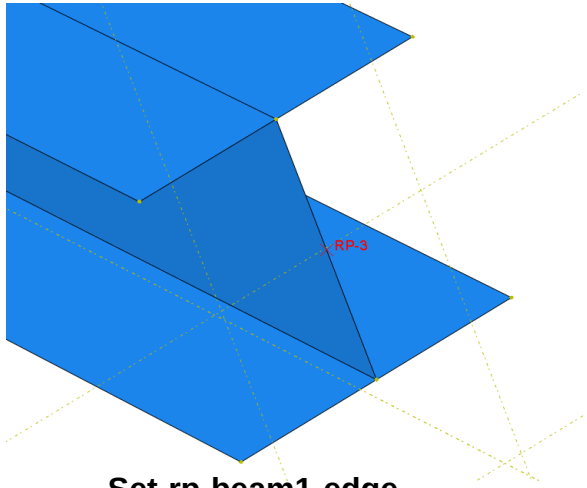




Set-section-support1



Datum csys-support1



Set-rp-beam1-edge

选取几何点，输入参考点 和 设置集合(set)

四.以倾斜钢管混凝土柱支托式斜撑为例

4.5 实现建模参数化

joint_type32_main.py

joint_type32_make_model.py

joint_type32_make_model_user.py

```
if 'geometry_beam2_1' in line:
    line = line.replace('geometry_beam2_1', str(list(geometry_beam[1, :])))
if 'geometry_beam3_1' in line:
    line = line.replace('geometry_beam3_1', str(list(geometry_beam[2, :])))
if 'geometry_beam4_1' in line:
    line = line.replace('geometry_beam4_1', str(list(geometry_beam[3, :])))
if 'material_property_1' in line:
    line = line.replace('material_property_1', str(material_property))
```

```
from joint_type32_make_model import *

if __name__ == '__main__':
    joint_example = joint26_model(geometry_column_1, geometry_plate_1, num_beams_1,
                                  geometry_beam1_1, geometry_beam2_1, geometry_beam3_1,
                                  geometry_beam4_1,num_supports, geometry_support_board,
                                support_compress_1,material_property_1, material_support_1,
                                 mesh_size_1, compute_set_1)

    joint_example.step1_part()
    joint_example.step2_assembly()
    joint_example.step3_property()
    joint_example.step4_mesh()
    joint_example.step5_RP()
    joint_example.step6_interation()
    joint_example.step7_step_out()
    joint_example.step8_load()
    joint_example.step9_job_submit()
```

四.以倾斜钢管混凝土柱支托式斜撑为例

4.6 读取结果文件并保存原始数据

场输出
(Field Output)

```
#场输出
regionPost = odb.rootAssembly.elementSets[setName]
stepNeed = odb.steps.keys()[-1]
frameNeed = odb.steps[str(stepNeed)].frames[int(i)]
miseseFieldOutputNeed = frameNeed.fieldOutputs['S']
postValue = miseseFieldOutputNeed.getSubset(region=regionPost).values
listValues = [element.mises for element in postValue]
```

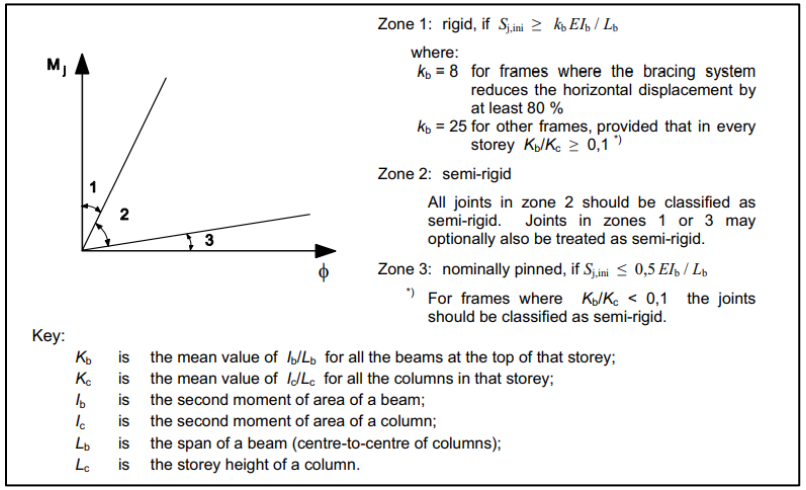
```
odbAccess.upgradeOdb(existingOdbPath=fileName,
                     upgradedOdbPath=newName)
odb = openOdb(newName + '.odb')
```

时程输出
(History Output)

```
#时程输出
step = odb.steps[odb.steps.keys()[-1]] #step1
historyPoint = step.historyRegions[dictName[node]] #Node ASSEMBLY.12
dataXy = historyPoint.historyOutputs[item].data  #RF3
```

四.以倾斜钢管混凝土柱支托式斜撑为例

4.7 根据所需写处理函数并保存处理结果



Zone 1: rigid, if $S_{j,ini} \geq k_b EI_b / L_b$

where:

$k_b = 8$ for frames where the bracing system reduces the horizontal displacement by at least 80 %

$k_b = 25$ for other frames, provided that in every storey $K_b/K_c \geq 0,1$ *)

Zone 2: semi-rigid

All joints in zone 2 should be classified as semi-rigid. Joints in zones 1 or 3 may optionally also be treated as semi-rigid.

Zone 3: nominally pinned, if $S_{j,ini} \leq 0,5 EI_b / L_b$

*) For frames where $K_b/K_c < 0,1$ the joints should be classified as semi-rigid.

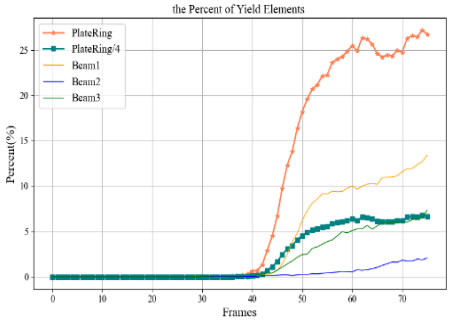
Key:

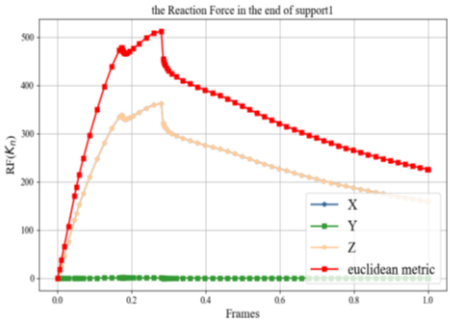
K_b	is	the mean value of I_b/L_b for all the beams at the top of that storey;
K_c	is	the mean value of I_c/L_c for all the columns in that storey;
I_b	is	the second moment of area of a beam;
I_c	is	the span of a beam (centre-to-centre of columns);
L_b	is	the storey height of a column.

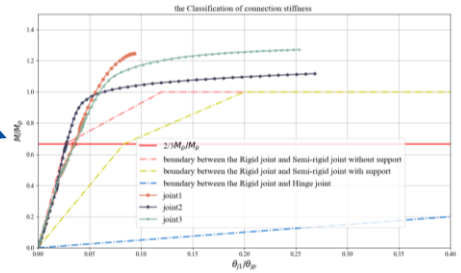
68：BS EN 1993-1-8：2005

- table_RF2_support1_edge.csv
- table_RF3_beam1_edge.csv
- table_RF3_beam2_edge.csv
- table_RF3_beam3_edge.csv
- table_RF3_beam4_edge.csv
- table_RF3_support1_edge.csv
- table_U1_beam1_down.csv
- table_U1_beam1_up.csv
- table_U1_beam2_down.csv

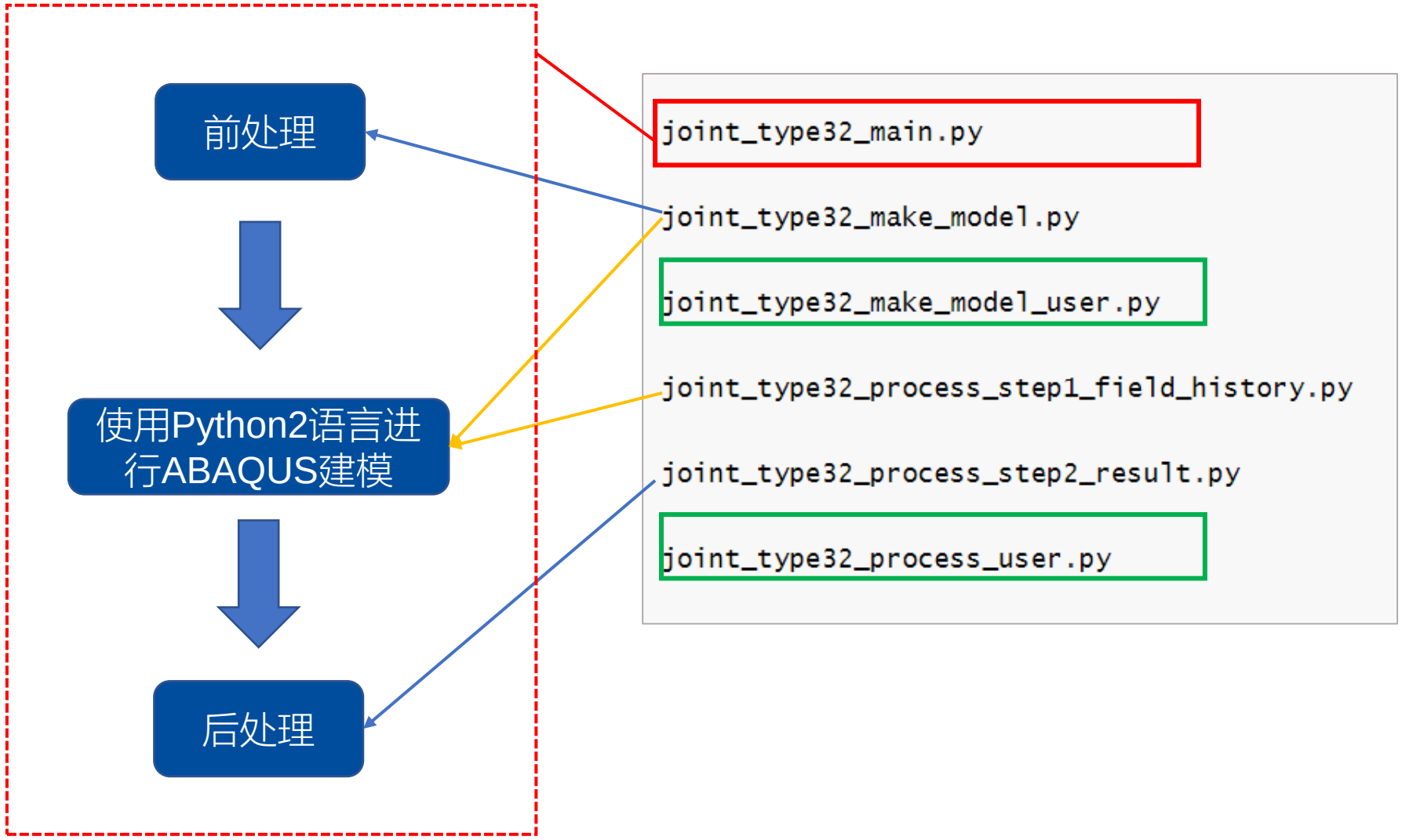
```
joint_type32_process_step2_result.py
...

def get_rf_picture(dict_process_data)...
#得到点集合的反力

def get_percent_picture(num_beams)...
#得到构件集合屈服单元的百分比

def show_BSEN_classification_of_connection_stifness()...
#判断节点类型
```

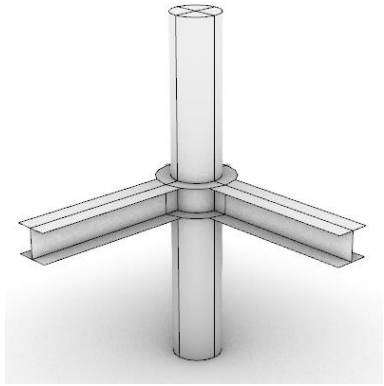
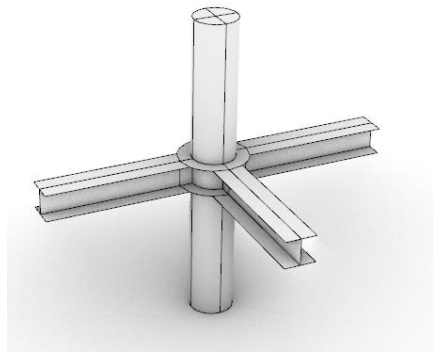
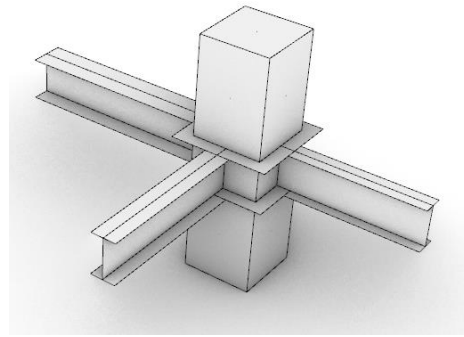
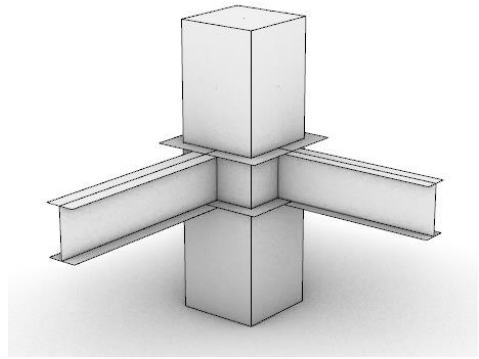
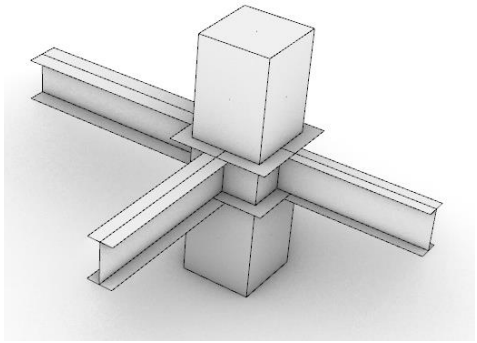
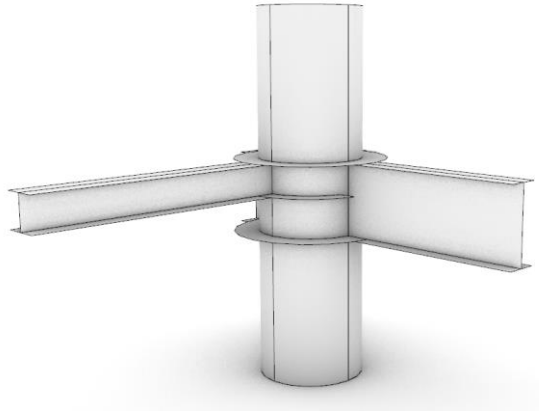
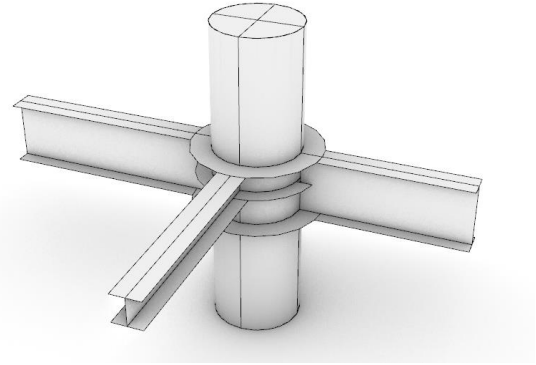
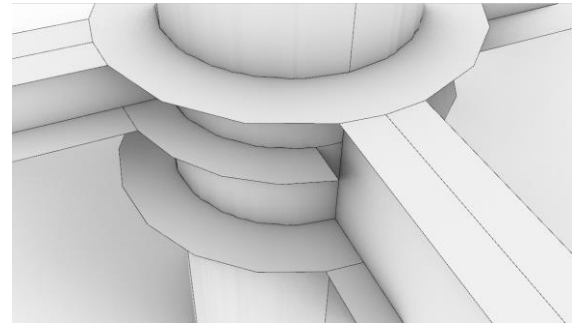
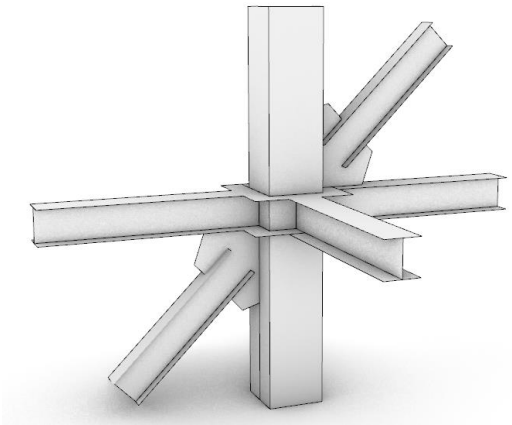
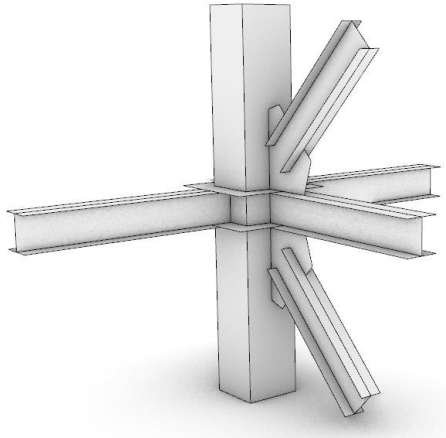
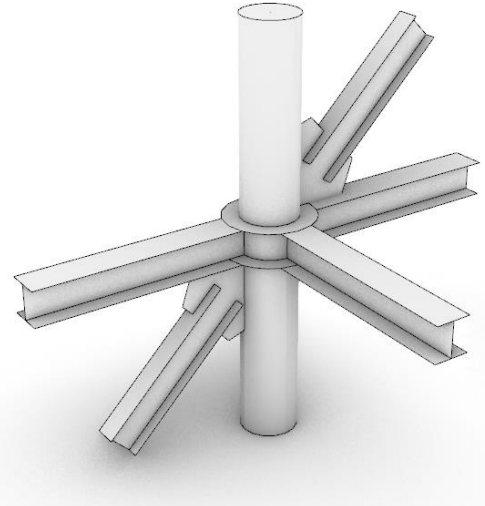
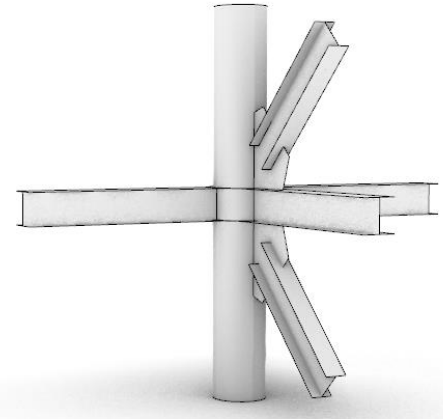






三. 参数化建模流程


前处理
使用Python2语言进行ABAQUS建模
后处理
joint_type32_main.py
joint_type32_make_model.py
joint_type32_make_model_user.py
joint_type32_process_step1_field_history.py
joint_type32_process_step2_result.py
joint_type32_process_user.py

欢迎各位老师和同学批评建议！