

**研究生课程作业**

课程名称： 有限元方法及应用

研究生姓名： 郝好顺 学号： S230200226

作业成绩：

任课教师(签名) 王琥

交作业时间：2024年2月 7 日

目录

[1 薄壁钢管在轴向压力作用下的屈曲分析 3](#_Toc158216053)

[1.1问题描述 3](#_Toc158216054)

[1.2理论分析 3](#_Toc158216055)

[2 薄壁钢管有限元分析的过程 4](#_Toc158216056)

[2.1有限元分析软件简介 4](#_Toc158216057)

[2.2前处理 4](#_Toc158216058)

[2.3提交计算 4](#_Toc158216059)

[2.4后处理 5](#_Toc158216060)

[2.5 精度对比 6](#_Toc158216061)

[3 Python部分 8](#_Toc158216062)

[3.1Python软件简介 8](#_Toc158216063)

[3.2代码部分（需要abaqus库） 8](#_Toc158216064)

薄壁钢管有限元屈曲分析

郝好顺

# 1 薄壁钢管在轴向压力作用下的屈曲分析

## 1.1问题描述

如图1所示的圆形钢管直径为28mm，壁厚为0.02mm，长度为80mm，杨氏模量2e11MPa，泊松比0.3。一端完全固定，另一端进行强制性位移加载，使其端面上受到500N的集中压力作用，分析钢管整个过程中的变形和应力分布。

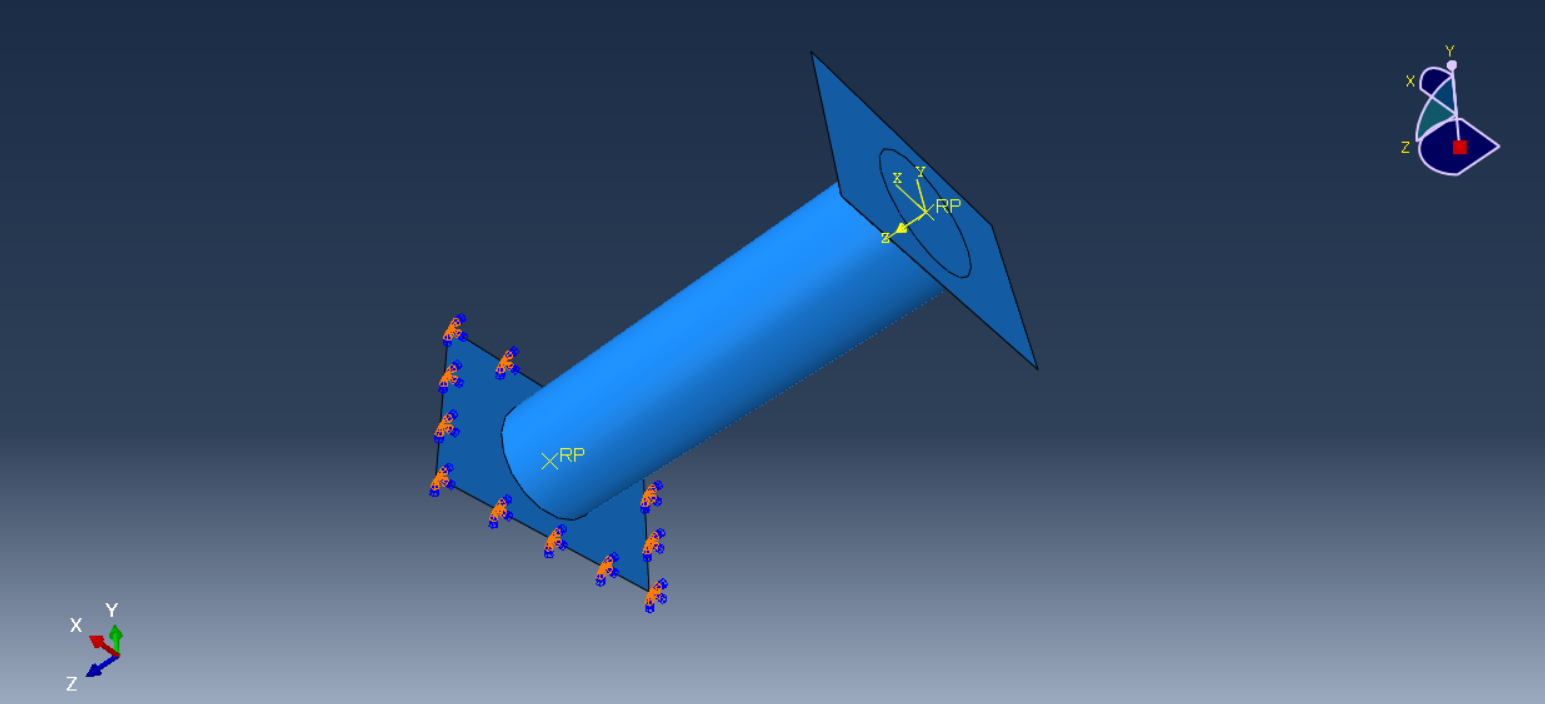


图1 薄壁钢管的结构模型

## 1.2理论分析

建立线性屈曲模型： 钢管在受力下会发生线性屈曲，因此可以使用欧拉屈曲理论。考虑在轴对称情况下，使用圆柱坐标系，制定相应的应力、应变关系。

边界条件： 一端完全固定，另一端进行强制性位移加载。这将导致钢管的轴向位移，并且在加载端面上将有一个集中压力作用。

数学建模： 将物理问题转化为数学问题，形成方程。在这个问题中，涉及到线性屈曲分析求解的方程：。其中是线弹性刚度矩阵；是特征值，荷载放大系数；是几何刚度矩阵；是特征向量，对应特征值的屈曲模态。

材料性质： 杨氏模量2e11MPa，泊松比0.3，确定材料的弹性模型。

加载条件： 一端面上受到500N的集中压力作用，作用点在钢管圆心处。

数值解法： 选择有限元方法。在有限元模型中，建立一个离散的网格，定义节点和元素，然后使用数值方法求解方程。对于线性屈曲分析，ABAQUS内部自动应用两种求解器进行求解：1.首先执行线性分析：2.基于静力学分析的基础上，计算应力刚度矩阵3.应用特征值方法求解得到屈曲载荷乘子和屈曲模态

# 2 薄壁钢管有限元分析的过程

## 2.1有限元分析软件简介

Abaqus是一款由达索系统公司打造的卓越有限元分析软件，旨在为工程师和科学家提供强大而灵活的工具，以解决各类工程和科学难题。其多领域模拟功能涵盖结构、热力学、电磁场等多个领域，使其成为处理复杂工程问题的首选工具。用户可以在Abaqus中精确定义几何形状、材料性质和加载条件，利用先进的有限元技术进行数值分析，从而获得可靠的模拟结果。此外，软件还提供强大的后处理功能，帮助用户可视化和分析模拟结果，深入了解系统行为。广泛应用于工程、航空航天、汽车和生物医学等领域，Abaqus在优化设计和解决工程挑战方面展现出无可匹敌的领导地位，推动着工程模拟的前沿。

## 2.2前处理

模型构建：

该圆筒直径为28mm，壁厚为0.02mm，长度为80mm，材料为结构钢，如图2所示。

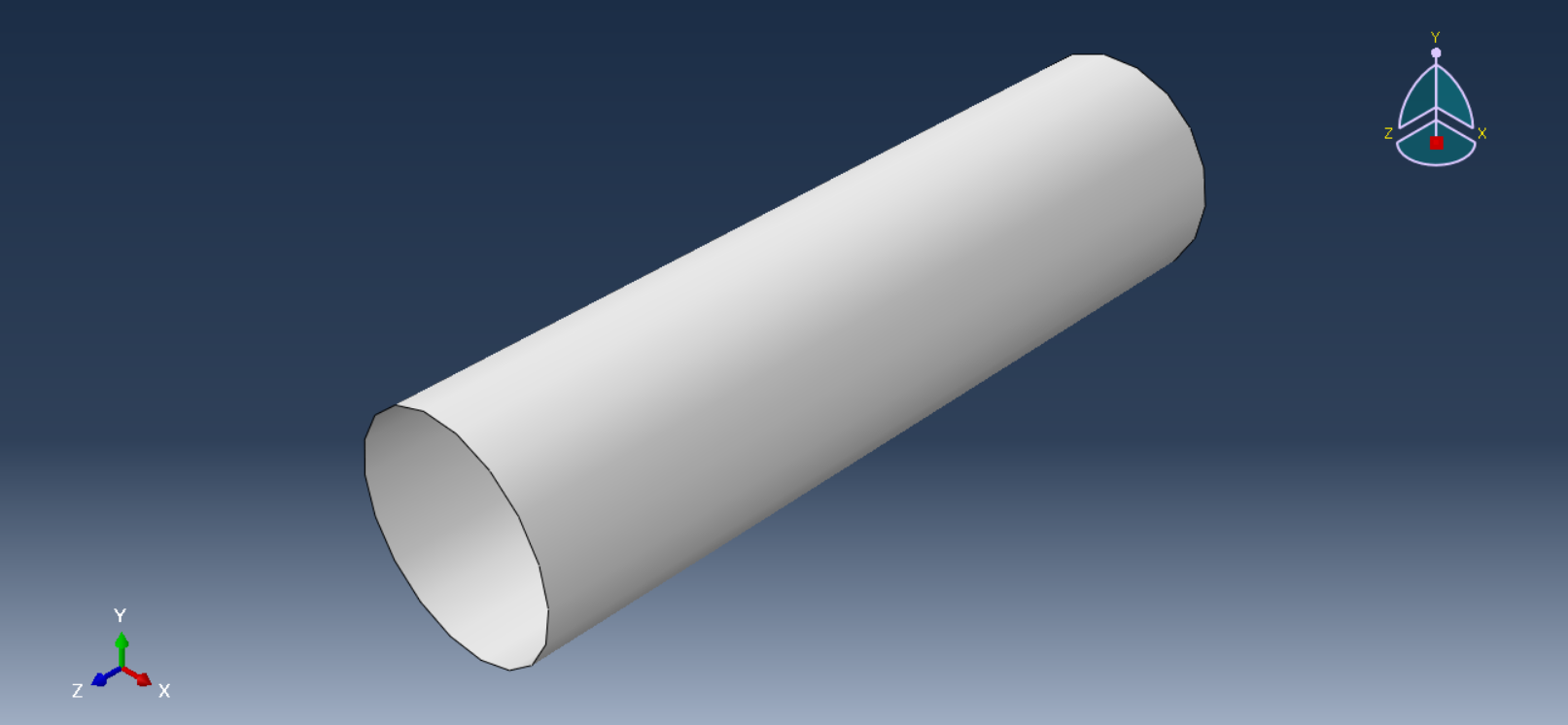


图2 圆筒模型构建

参数定义：杨氏模量2e11MPa，泊松比0.3。

分析步：选择线性摄动：屈曲分析步，请求特征值个数为10，用在每次迭代的向量是18，最大迭代次数为50。

载荷：左端固定，右端施加500N集中力。

网格划分：采用四面体单元，结构单元类型为S4R。

## 2.3提交计算

创建作业文件Job-1，提交计算。

## 2.4后处理

如表1所示，给出了前五阶屈曲分析结果：

表1 前五阶屈曲分析结果

|  |  |
| --- | --- |
| 1阶屈曲结果： |  |
| 2阶屈曲结果： |  |
| 3阶屈曲结果： |  |
| 4阶屈曲结果： |  |
| 5阶屈曲结果： |  |

## 2.5 精度对比

上述分析结果是基于划分了1600个单元，下面将单元数量改为2850，比较其结果。如表2所示。

表2 多单元数量前五阶屈曲结果

|  |  |
| --- | --- |
| 1阶屈曲结果： |  |
| 2阶屈曲结果： |  |
| 3阶屈曲结果： |  |
| 4阶屈曲结果： |  |
| 5阶屈曲结果： |  |

可以看出，2850单元数量的5阶屈曲结果和1600单元数量的4阶屈曲结果很相似，即多单元数量的高阶屈曲结果和少单元数量的低阶屈曲结果差不多。说明了更多的单元可以提高模型的准确性，即能更细致地捕捉结构的几何形状和变形。但是也需要注意到，单元过多会导致计算量增加，从而增加计算时间。

# 3 Python部分

## 3.1Python软件简介

Python是一种高级、通用、解释型编程语言，具有简洁、易读的语法，以及强大的标准库。它由Guido van Rossum在1989年创立，被设计为易学易用的语言，强调代码的可读性和清晰性。Python支持面向对象、过程式和函数式编程范式，使得程序员能够选择最适合其需求的编程风格。Python在众多领域中得到广泛应用，包括Web开发、数据科学、人工智能、机器学习、自然语言处理等。其丰富的第三方库和框架，如Django、Flask、NumPy、Pandas等，极大地提高了开发效率。由于其跨平台性，Python成为了一个流行的编程语言，无论是初学者还是经验丰富的开发者都能够从中受益。

Python解释器易于扩展，可以使用C语言或C++（或者其他可以通过C调用的语言）扩展新的功能和数据类型。Python也可用于可定制化软件中的扩展程序语言。Python丰富的标准库，提供了适用于各个主要系统平台的源码或机器码。

## 3.2代码部分（需要abaqus库）

# -\*- coding: mbcs -\*-

# This Python file uses the following encoding: utf-8

# Abaqus/CAE Release 6.14-4 replay file

# Internal Version: 2015\_06\_12-04.41.13 135079

#

# from driverUtils import executeOnCaeGraphicsStartup

# executeOnCaeGraphicsStartup()

#: Executing "onCaeGraphicsStartup()" in the site directory ...

from abaqus import \*

from abaqusConstants import \*

#==========create model==========

Mdb

session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=217.015625,

height=131.570602416992)

session.viewports['Viewport: 1'].makeCurrent()

session.viewports['Viewport: 1'].maximize()

from caeModules import \*

from driverUtils import executeOnCaeStartup

executeOnCaeStartup()

session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(

referenceRepresentation=ON)

s = mdb.models['Model-1'].ConstrainedSketch(name='\_\_profile\_\_',

sheetSize=200.0)

g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints

s.setPrimaryObject(option=STANDALONE)

s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(14.0, 0.0))

p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=THREE\_D,

type=DEFORMABLE\_BODY)

p = mdb.models['Model-1'].parts['Part-1']

p.BaseShellExtrude(sketch=s, depth=100.0)

s.unsetPrimaryObject()

p = mdb.models['Model-1'].parts['Part-1']

session.viewports['Viewport: 1'].setValues(displayedObject=p)

del mdb.models['Model-1'].sketches['\_\_profile\_\_']

s1 = mdb.models['Model-1'].ConstrainedSketch(name='\_\_profile\_\_',

sheetSize=200.0)

g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints

s1.setPrimaryObject(option=STANDALONE)

s1.rectangle(point1=(-30.0, 20.0), point2=(30.0, -20.0))

p = mdb.models['Model-1'].Part(name='Part-2', dimensionality=THREE\_D,

type=DEFORMABLE\_BODY)

p = mdb.models['Model-1'].parts['Part-2']

p.BaseShell(sketch=s1)

s1.unsetPrimaryObject()

p = mdb.models['Model-1'].parts['Part-2']

session.viewports['Viewport: 1'].setValues(displayedObject=p)

del mdb.models['Model-1'].sketches['\_\_profile\_\_']

p = mdb.models['Model-1'].parts['Part-2']

p.ReferencePoint(point=(0.0, 0.0, 0.0))

p1 = mdb.models['Model-1'].parts['Part-2']

session.viewports['Viewport: 1'].setValues(displayedObject=p1)

p = mdb.models['Model-1'].Part(name='Part-3',

objectToCopy=mdb.models['Model-1'].parts['Part-2'])

session.viewports['Viewport: 1'].setValues(displayedObject=p)

session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,

engineeringFeatures=ON)

session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(

referenceRepresentation=OFF)

p = mdb.models['Model-1'].parts['Part-1']

session.viewports['Viewport: 1'].setValues(displayedObject=p)

#==========define material==========

mdb.models['Model-1'].Material(name='Material-1')

mdb.models['Model-1'].materials['Material-1'].Elastic(table=((200000000000.0,

0.3), ))

mdb.models['Model-1'].HomogeneousShellSection(name='Section-1',

preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,

thickness=0.02, thicknessField='', idealization=NO\_IDEALIZATION,

poissonDefinition=DEFAULT, thicknessModulus=None, temperature=GRADIENT,

useDensity=OFF, integrationRule=SIMPSON, numIntPts=3)

p = mdb.models['Model-1'].parts['Part-1']

f = p.faces

faces = f.getSequenceFromMask(mask=('[#1 ]', ), )

region = regionToolset.Region(faces=faces)

p = mdb.models['Model-1'].parts['Part-1']

p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,

offsetType=MIDDLE\_SURFACE, offsetField='',

thicknessAssignment=FROM\_SECTION)

p = mdb.models['Model-1'].parts['Part-2']

session.viewports['Viewport: 1'].setValues(displayedObject=p)

p = mdb.models['Model-1'].parts['Part-2']

f = p.faces

faces = f.getSequenceFromMask(mask=('[#1 ]', ), )

region = regionToolset.Region(faces=faces)

p = mdb.models['Model-1'].parts['Part-2']

p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,

offsetType=MIDDLE\_SURFACE, offsetField='',

thicknessAssignment=FROM\_SECTION)

p = mdb.models['Model-1'].parts['Part-3']

session.viewports['Viewport: 1'].setValues(displayedObject=p)

p = mdb.models['Model-1'].parts['Part-3']

f = p.faces

faces = f.getSequenceFromMask(mask=('[#1 ]', ), )

region = regionToolset.Region(faces=faces)

p = mdb.models['Model-1'].parts['Part-3']

p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,

offsetType=MIDDLE\_SURFACE, offsetField='',

thicknessAssignment=FROM\_SECTION)

#==========create steps==========

a = mdb.models['Model-1'].rootAssembly

session.viewports['Viewport: 1'].setValues(displayedObject=a)

session.viewports['Viewport: 1'].assemblyDisplay.setValues(

optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)

a = mdb.models['Model-1'].rootAssembly

a.DatumCsysByDefault(CARTESIAN)

p = mdb.models['Model-1'].parts['Part-1']

a.Instance(name='Part-1-1', part=p, dependent=OFF)

p = mdb.models['Model-1'].parts['Part-2']

a.Instance(name='Part-2-1', part=p, dependent=OFF)

p = mdb.models['Model-1'].parts['Part-3']

a.Instance(name='Part-3-1', part=p, dependent=OFF)

a = mdb.models['Model-1'].rootAssembly

1. translate(instanceList=('Part-3-1', ), vector=(0.0, 0.0, 100.0))

#: The instance Part-3-1 was translated by 0., 0., 100. (相对于装配坐标系)

session.viewports['Viewport: 1'].assemblyDisplay.setValues(

adaptiveMeshConstraints=ON)

mdb.models['Model-1'].BuckleStep(name='Step-1', previous='Initial',

numEigen=10, vectors=18, maxIterations=50)

session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Step-1')

#========== create interaction==========

mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(

'U', 'UT', 'UR', 'RBANG', 'RBROT'))

session.viewports['Viewport: 1'].assemblyDisplay.setValues(interactions=ON,

constraints=ON, connectors=ON, engineeringFeatures=ON,

adaptiveMeshConstraints=OFF)

mdb.models['Model-1'].ContactProperty('IntProp-1')

mdb.models['Model-1'].interactionProperties['IntProp-1'].TangentialBehavior(

formulation=FRICTIONLESS)

mdb.models['Model-1'].interactionProperties['IntProp-1'].NormalBehavior(

pressureOverclosure=HARD, allowSeparation=ON,

constraintEnforcementMethod=DEFAULT)

#: 相互作用属性 "IntProp-1" 已创建.

session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Initial')

a = mdb.models['Model-1'].rootAssembly

s1 = a.instances['Part-2-1'].faces

side2Faces1 = s1.getSequenceFromMask(mask=('[#1 ]', ), )

region1=regionToolset.Region(side2Faces=side2Faces1)

a = mdb.models['Model-1'].rootAssembly

e1 = a.instances['Part-1-1'].edges

edges1 = e1.getSequenceFromMask(mask=('[#2 ]', ), )

region2=regionToolset.Region(edges=edges1)

mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='Int-1',

createStepName='Initial', master=region1, slave=region2, sliding=FINITE,

enforcement=NODE\_TO\_SURFACE, thickness=OFF,

interactionProperty='IntProp-1', surfaceSmoothing=NONE, adjustMethod=NONE,

smooth=0.2, initialClearance=OMIT, datumAxis=None, clearanceRegion=None)

#: 相互作用 "Int-1" 已创建.

a = mdb.models['Model-1'].rootAssembly

s1 = a.instances['Part-3-1'].faces

side1Faces1 = s1.getSequenceFromMask(mask=('[#1 ]', ), )

region1=regionToolset.Region(side1Faces=side1Faces1)

a = mdb.models['Model-1'].rootAssembly

e1 = a.instances['Part-1-1'].edges

edges1 = e1.getSequenceFromMask(mask=('[#1 ]', ), )

region2=regionToolset.Region(edges=edges1)

mdb.models['Model-1'].Tie(name='Constraint-1', master=region1, slave=region2,

positionToleranceMethod=COMPUTED, adjust=ON, tieRotations=ON, thickness=ON)

session.viewports['Viewport: 1'].view.setValues(nearPlane=183.457,

farPlane=337.817, width=160.059, height=71.1316, cameraPosition=(71.6748,

188.617, -114.992), cameraUpVector=(-0.687567, -0.583842, -0.431718))

session.viewports['Viewport: 1'].view.setValues(nearPlane=175.3,

farPlane=345.973, width=259.01, height=115.106, viewOffsetX=9.78767,

viewOffsetY=-2.47856)

a = mdb.models['Model-1'].rootAssembly

f1 = a.instances['Part-2-1'].faces

faces1 = f1.getSequenceFromMask(mask=('[#1 ]', ), )

region2=regionToolset.Region(faces=faces1)

a = mdb.models['Model-1'].rootAssembly

r1 = a.instances['Part-2-1'].referencePoints

refPoints1=(r1[2], )

region1=regionToolset.Region(referencePoints=refPoints1)

mdb.models['Model-1'].RigidBody(name='Constraint-2', refPointRegion=region1,

bodyRegion=region2)

#==========load==========

session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON, bcs=ON,

predefinedFields=ON, interactions=OFF, constraints=OFF,

engineeringFeatures=OFF)

a = mdb.models['Model-1'].rootAssembly

f1 = a.instances['Part-2-1'].faces

faces1 = f1.getSequenceFromMask(mask=('[#1 ]', ), )

region = regionToolset.Region(faces=faces1)

mdb.models['Model-1'].DisplacementBC(name='BC-1', createStepName='Initial',

region=region, u1=SET, u2=SET, u3=UNSET, ur1=SET, ur2=SET, ur3=SET,

amplitude=UNSET, distributionType=UNIFORM, fieldName='', localCsys=None)

del mdb.models['Model-1'].boundaryConditions['BC-1']

a = mdb.models['Model-1'].rootAssembly

f1 = a.instances['Part-3-1'].faces

faces1 = f1.getSequenceFromMask(mask=('[#1 ]', ), )

region = regionToolset.Region(faces=faces1)

mdb.models['Model-1'].EncastreBC(name='BC-1', createStepName='Initial',

region=region, localCsys=None)

session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Step-1')

a = mdb.models['Model-1'].rootAssembly

r1 = a.instances['Part-2-1'].referencePoints

refPoints1=(r1[2], )

region = regionToolset.Region(referencePoints=refPoints1)

mdb.models['Model-1'].ConcentratedForce(name='Load-1', createStepName='Step-1',

region=region, cf3=500.0, distributionType=UNIFORM, field='',

localCsys=None)

#==========mesh=========

session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON, loads=OFF,

bcs=OFF, predefinedFields=OFF, connectors=OFF)

session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(

meshTechnique=ON)

a = mdb.models['Model-1'].rootAssembly

e1 = a.instances['Part-1-1'].edges

e2 = a.instances['Part-2-1'].edges

e3 = a.instances['Part-3-1'].edges

pickedEdges = e1.getSequenceFromMask(mask=('[#3 ]', ), )+\

e2.getSequenceFromMask(mask=('[#f ]', ), )+e3.getSequenceFromMask(mask=(

'[#f ]', ), )

a.seedEdgeByNumber(edges=pickedEdges, number=25, constraint=FINER)

a = mdb.models['Model-1'].rootAssembly

e1 = a.instances['Part-1-1'].edges

e2 = a.instances['Part-2-1'].edges

e3 = a.instances['Part-3-1'].edges

pickedEdges = e1.getSequenceFromMask(mask=('[#3 ]', ), )+\

e2.getSequenceFromMask(mask=('[#f ]', ), )+e3.getSequenceFromMask(mask=(

'[#f ]', ), )

a.seedEdgeByNumber(edges=pickedEdges, number=5, constraint=FINER)

a = mdb.models['Model-1'].rootAssembly

e1 = a.instances['Part-2-1'].edges

e2 = a.instances['Part-3-1'].edges

pickedEdges = e1.getSequenceFromMask(mask=('[#f ]', ), )+\

e2.getSequenceFromMask(mask=('[#f ]', ), )

a.seedEdgeByNumber(edges=pickedEdges, number=10, constraint=FINER)

a = mdb.models['Model-1'].rootAssembly

e1 = a.instances['Part-1-1'].edges

pickedEdges = e1.getSequenceFromMask(mask=('[#3 ]', ), )

a.seedEdgeByNumber(edges=pickedEdges, number=40, constraint=FINER)

elemType1 = mesh.ElemType(elemCode=S4R, elemLibrary=STANDARD,

secondOrderAccuracy=OFF, hourglassControl=DEFAULT)

elemType2 = mesh.ElemType(elemCode=S3, elemLibrary=STANDARD)

a = mdb.models['Model-1'].rootAssembly

f1 = a.instances['Part-1-1'].faces

faces1 = f1.getSequenceFromMask(mask=('[#1 ]', ), )

f2 = a.instances['Part-2-1'].faces

faces2 = f2.getSequenceFromMask(mask=('[#1 ]', ), )

f3 = a.instances['Part-3-1'].faces

faces3 = f3.getSequenceFromMask(mask=('[#1 ]', ), )

pickedRegions =((faces1+faces2+faces3), )

a.setElementType(regions=pickedRegions, elemTypes=(elemType1, elemType2))

a = mdb.models['Model-1'].rootAssembly

partInstances =(a.instances['Part-1-1'], a.instances['Part-2-1'],

a.instances['Part-3-1'], )

a.generateMesh(regions=partInstances)

session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=OFF)

session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(

meshTechnique=OFF)

mdb.Job(name='Job-1', model='Model-1', description='', type=ANALYSIS,

atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,

memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,

explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,

modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',

scratch='', resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=1,

numGPUs=0)

#==========create job and submit job=========

mdb.jobs['Job-1'].submit(consistencyChecking=OFF)

#: 作业输入文件 "Job-1.inp" 已提交分析.

#: Job Job-1: Analysis Input File Processor completed successfully.

#: Job Job-1: Abaqus/Standard completed successfully.

#: Job Job-1 completed successfully.

o3 = session.openOdb(name='C:/Temp/Job-1.odb')

#: 模型: C:/Temp/Job-1.odb

#: 装配件个数: 1

#: 装配件实例个数: 0

#: 部件实例的个数: 3

#: 网格数: 3

#: 单元集合数: 1

#: 结点集合数: 2

#: 分析步的个数: 1

session.viewports['Viewport: 1'].setValues(displayedObject=o3)

session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(

CONTOURS\_ON\_DEF, ))

session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=2 )

session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=3 )

session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=4 )

session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=5 )