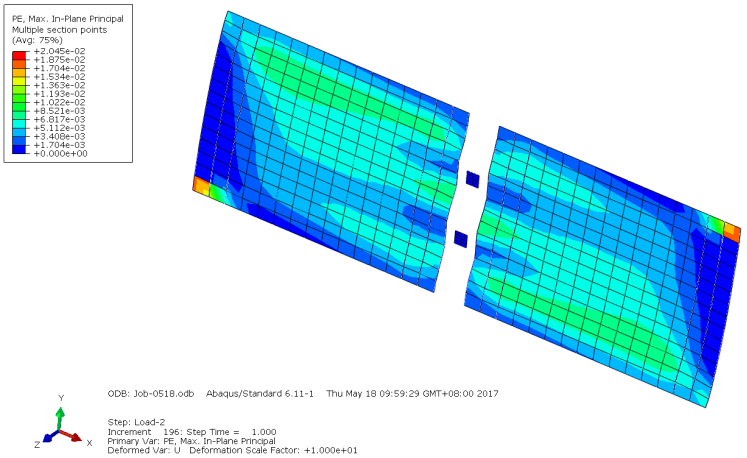
清華大學 Tsinghua University



基于 PYTHON 的 ABAQUS 二次开发研究

课程大作业总结

信息

土硕 16　2016210135

龚润华

土木与建筑工程 CAE

基于 Python 的 ABAQUS 二次开发研究总结报告

龚润华

0. 前言

ABAQUS 作为大型通用有限元软件，由于其强大的非线性计算功能与丰富的后处理功能，在有限元分析软件中逐渐占据了更大的市场份额和更重要的地位。在建筑结构中，广泛应用到抗震设计当中，特别是对于在大震作用下的弹塑性响应分析。而相比于更为专业的建筑结构设计软件，如 PKPM、Etabs，则更关注于建筑结构的高效设计，在建筑结构建模、弹性分析等方面做得较好，但在弹塑性分析中则无法凸显出优势来。虽然 ABAQUS 有着诸多优势，但在建筑设计中的一大问题是，没有“标准层”的概念，这导致在设计建模分析当中工作量较大，难以快速组装成建筑结构。

考虑到这个问题对于实际工程设计存在一定的促进意义，同时我也可以在研究、解决这个问题的过程当中，对建筑结构设计流程、有限元软件 ABAQUS 获得进一步的深入了解，因此选择这个问题作为课程大作业。考虑到需要基于 ABAQUS 这个有限元软件平台解决这个问题，因此考虑对其进行二次开发，而针对 ABAQUS 而言，常用的两种开发语言分别是 C++和 Python，其中 ABAQUS 为 Python 提供了脚本接口和窗口命令行接口，并且 ABAQUS 内部编码也是由 Python 语言实现的，同时考虑到 Python 语言更为简练和通用，因此最终确定研究目标为基于 Python 对 ABAQUS 进行二次开发，解决 ABAQUS 中没有建筑结构设计当中常用的“标准层”的概念。

总结报告共分为六个部分：一、选题背景；二、研究目的与内容；三、实现路径；四、研究成果；五、心得体会；六、对于课程的建议。

一、 选题背景与目的

ABAQUS 作为大型通用有限元分析软件，自 1978 年由 HKS 公司发布，标志着 ABAQUS 进入市场，其强大的非线性分析能力就对行业带来实质性的冲击。ABAQUS 主要可进行以下几个方面的分析工作：

1) 静态应力、位移分析
2) 动态弹塑性分析
3) 非线性动态应力位移分析
4) 瞬态温度位移耦合分析
5) 疲劳分析
6) ……

ABAQUS 在前处理过程中，不仅提供了用户友好的交互设计 CAE 界面，还提供了丰富的隐式或显式积分求解方法；在后处理过程中，提供了多样化的数据处理和呈现方式，能够更加综合、直观的表达数据结果。

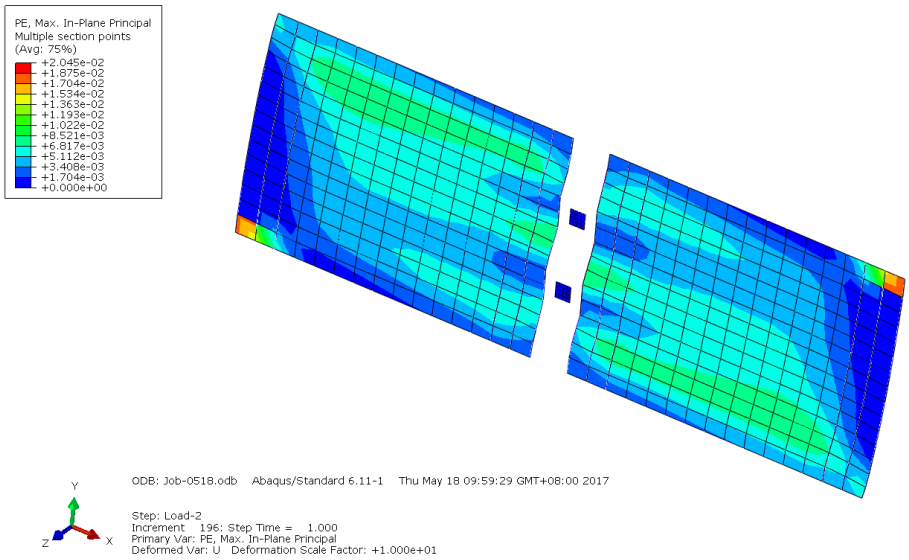


图 1 典型的 ABAQUS 分析结果云图

随着中国经济、文化的快速发展，日益增长的人口和人们对于居住城市的发展水平和住宅舒适程度逐步提升，高层住宅在发达城市的需求日益旺盛。随着对高层住宅需求的日益增长，对于高层住宅结构的设计也成为了越来越重要的话题。一般而言，高层民用住宅多为剪力墙体系或框架剪力墙体系，而在现行设计规范，如《建筑抗震设计规范》（GB 50011-2010）中则对超出规范限值的建筑结构，要求在设计中补充大震作用下的弹塑性验算。而一般在设计中，设计人员往往会选用分析功能强大、后处理功能丰富的通用有限元软件 ABAQUS 作为建筑结构的弹塑性分析工具。但由于一般的转用设计软件，例如 PKPM、盈建科、Etabs，均没有通用的模型转换工具，能直接将设计模型转换为可用于 ABAQUS 的分析模型，因此一般而言设计人员均需要进行重新建模或者采用自行设计的模型转换接口进行模型转换，费时而且对于复杂模型容易出错需要调试。因此，研究采用何种方式或何种路径，能够更方便地在 ABAQUS 中进行建筑结构，特别是高层民用住宅的建模，是一件非常有意义的事情。

进一步分析高层民用住宅结构体系的具体特点，如图 2 为一典型高层民用住宅结构的有限元模型。通过总结，可以有如下结论：

1) 使用空间和目标高度一定
2) 平面布置单一化，上下贯通，一般只是将局部或者构件尺寸进行修改
3) 较少出现错层，一般情况下各层竖向构件高度一致
4) 层与层之间一般均为共节点，不存在复杂的连接构造

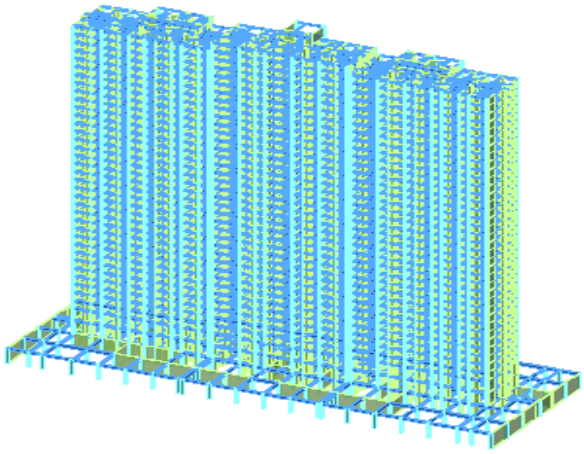


图 2 典型高层民用住宅结构有限元模型

为了解决这个问题，需要进一步对比通用有限元分析软件 ABAQUS 和专业建筑结构设计软件 PKPM。具体的对比结果总结如下表 1：

表 1 ABAQUS 与 PKPM 特点对比

	ABAQUS	PKPM
软件类别	通用有限元分析软件	专业建筑结构设计软件
专长领域	非线性分析	建模、线性分析
建模方式	Part→instance→assembly	层→整体结构
建模效率	较低	较高
分析结果	较为丰富，有可视化	较为单一，无可视化

根据以上分析可知，两款软件有着不同的适用领域：PKPM 更侧重于建筑结构的平面布局设计以及弹性分析，并且与工程后续工作有紧密的配合；ABAQUS 则更侧重于进行建筑结构的弹塑性分析，相对比较独立，而且不是专门面向建筑结构的有限元分析工具，因此没有侧重建筑结构的建模方法。而在建筑结构的设计中，最重要的差别之一是 PKPM 中有“标准层”的概念，并且计算结果中给出了楼层相关的参数，能够方便设计人员快速的检验是否满足设计要求，这与建筑结构特别是高层民用住宅在高程方向的单一性是相契合的，这在一定程度上省去了许多重复性的工作，提高了建模分析的效率。因此，如何在 ABAQUS 中引入“标

准层”的概念，提高在 ABAQUS 中的建模效率，是核心问题。一般而言，提高建模效率的主要方法是将重复性的工作通过编程实现自动化，而 ABAQUS 给用户们提供了很好的二次开发的平台和空间，图 3 为进行通过 Python 进行 ABAQUS 二次开发的主要途径，包括 GUI 接口、命令行接口和脚本接口。

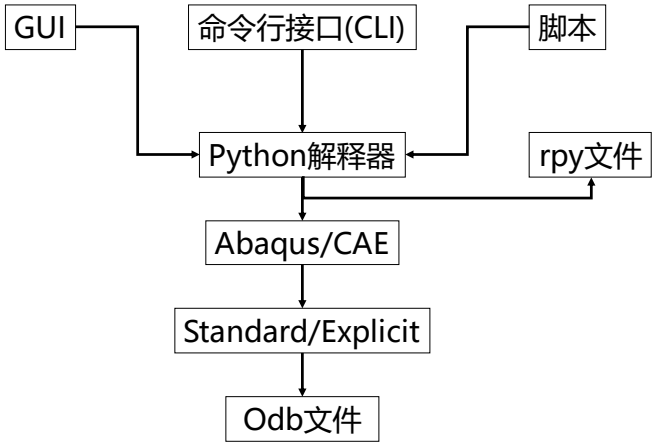


图 3 基于 Python 进行 ABAQUS 二次开发主要途径

二、 研究目的与内容

通过上述分析，可知从需求和技术层面来看，在 ABAQUS 中引入“标准层”是有意义并且可行的。那么具体而言，本次课程大作业的研究工作主要目标包括以下几个方面：

1) 以 ABAQUS 软件为基础，Python 为开发工具，引入“标准层”的概念，进行 ABAQUS 的二次开发
2) 以“标准层”的概念为基础，将原有 ABAQUS 中建模的自由度进行缩减，提出更适合于建筑结构，特别是高层民用住宅结构的建模思路
3) 基于“标准层”的概念，提出更为综合的指标参数，供设计人员参考
4) 二次开发的成果具有一定的开放性，可供不同的需求功能的接入

建筑结构的大震弹塑性分析是一个综合分析的过程，设计人员需要对数据结果进行综合分析，才能得到对于建筑结构的抗震性能以及是否满足规范要求的评价。而从数据的直观可操作性和重要性来看，大震弹塑性分析的结果中，弹塑性层间位移角是最为重要的评价标准之一，而且也方便进行参数化设计。因此本研究的主要内容可概括为：基于 Python 对 ABAQUS 进行二次开发，引入“标准层”的概念，在 ABAQUS 中实现部分参数化建模，并以弹塑性层间位移角为目标分析参数，实现可视化，并进一步以最优弹塑性层间位移角为目标，进行参数优化设计。

三、 实现路径

针对第二章中提出的具体研究目标，将具体的实现路径分为以下三个步骤：一、解析ABAQUS 模型信息与“标准层”概念引入方法；二、明确程序实现功能的过程；三、进行实现过程中各个功能模块的目标细化，并进行编程开发。

3.1. 模型信息选取

ABAQUS 有着多样化的数据文件，具体可见下表 2：

表 2 ABAQUS 中不同的数据文件

Session 对象	Abaqus 会话中的对象
Mdb 对象	Abaqus 有限元模型的根对象，包含两个成员对象 Model 对象和 Job 对象
Odb 对象	与 Mdb 对象类似，但用来记录分析结果数据

除了上述三种常见的数据文件类型外，还有进行前处理中的常用的模型信息文件——inp 文件，以及数据结果的存储文件 dat、msg 文件。针对希望在 ABAQUS 中引入“标准层”这个目标，对于模型的操作过程如下图 4 所示：

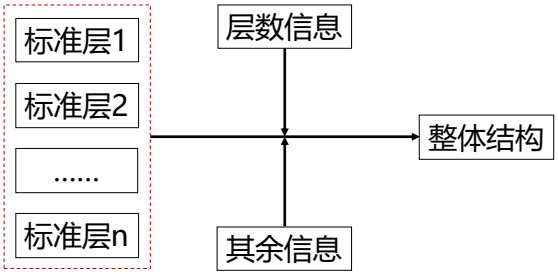


图 4 模型信息操作流程

考虑到针对不同的标准层而言，主要的空间布置均相同，主要差别在于材料属性和截面尺寸的差别，另外，对于一个标准层的模型文件而言，需要的是前处理建立的模型信息，并且一般来说比较简单而且还需要有较好的可更改性，方便设计人员进行实时调整。因此，综合考虑上述因素，最后选用前处理中常用的模型信息文件——inp 文件作为输入的模型信息文件类型。inp 文件信息易于读取，层次清晰，在程序实现中将利用提取关键词的位置来分段提取对应的模型信息。

3.2. 程序功能实现过程

选定了进行操作的目标文件类型后，需建立起程序功能的整体框架，具体如图 5 所示。

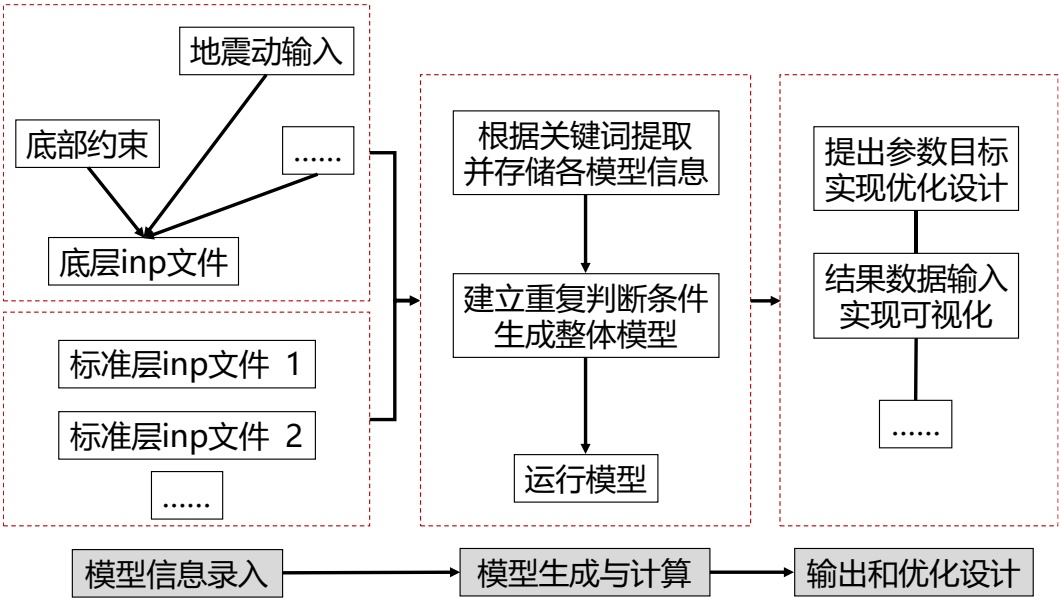


图5 程序功能整体框架

首先是对于输入模型信息的要求。由于最终的分析目标是进行大震下的弹塑性时程分析，因此需要在结构基础部位输入地震动，同时建立边界条件进行约束。因此，底部层的模型文件比上部标准层的模型文件信息更加丰富。另外还需考虑到模型信息的其余部分，例如分析步的导入，因此需要预留其余模型信息的导入接口。

其次是针对导入的不同标准层的模型进行整合。在除了读取不同标准层 inp 文件中的模型信息之外，还需读取用户输入的配套信息例如各个标准层的数量和组装关系。基于这些信息对不同标准层进行组装，在组装过程中主要需要解决的问题是对于重合点的判断和合并，并基于点的合并对点集、单元、单元集等等信息进行合并处理。

最后需要对整合后的模型 inp 文件进行运行，并提取层间位移角结果，对其进行可视化处理，并进行目标优化设计功能的开发。

3.3. 模块功能开发

基于 3.2 节中的程序框架，针对不同的功能模块提出具体的功能实现：

表 3 不同功能模块的具体功能实现

输出功能模块
➢ 读取不同标准层的 inp 文件以及可能存在的自定义 inp 文件 ➢ 读其余的模型定义信息 ➢ 输出分析结果至文档保存
模型信息整合模块
➢ 对读取的模型沿高程方向进行拼装，建立判断条件（<0.0001），合并重合点
运行分析模块
➢ 运行生成的整体模型 inp 文件 ➢ 显示操作过程状态，实现用户友好
后处理模块
➢ 提取生成的计算结果 odb 文件中的层间位移角 ➢ 生成可视化图像
优化设计模块
➢ 给定目标层数和目标层间位移角，实现不同标准层数量组合的优化设计

四、 研究成果

4.1. 程序组件

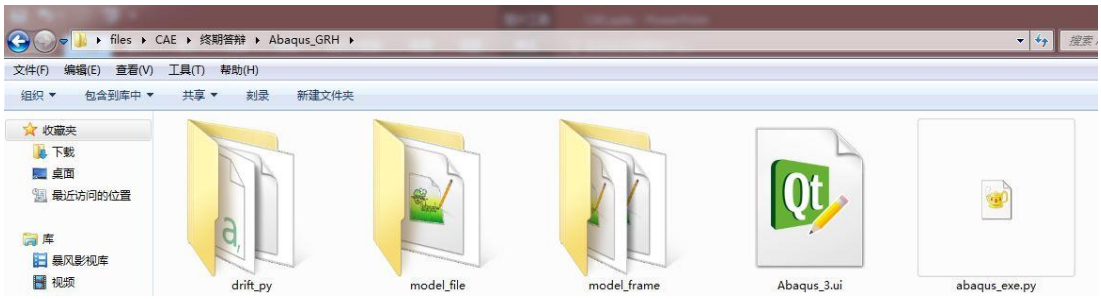


图 6 程序组件

程序各组件如上图 6 所示。具体各组件的功能以及开发环境如下表 4。

表 4 程序组件的功能

编程环境	Python2.7　ABAQUS6.11
第三方库	pyqt4（GUI 界面设计）　pyinstaller（py 文件转 exe 文件）
Drift_py	内含 drift.py 脚本，调用 Abaqus 环境对生成的结果 odb 文件提取数据并生成层间位移角
Model_file	整合生成的模型 inp 文件及运行结果文件所在的文件夹
Model_frame	输入的标准层 inp 文件及其它 inp 文件所在文件夹
Abaqus_exe.py	主程序脚本文件
Abaqus.ui	界面设计文件

4.2. 主程序窗口界面

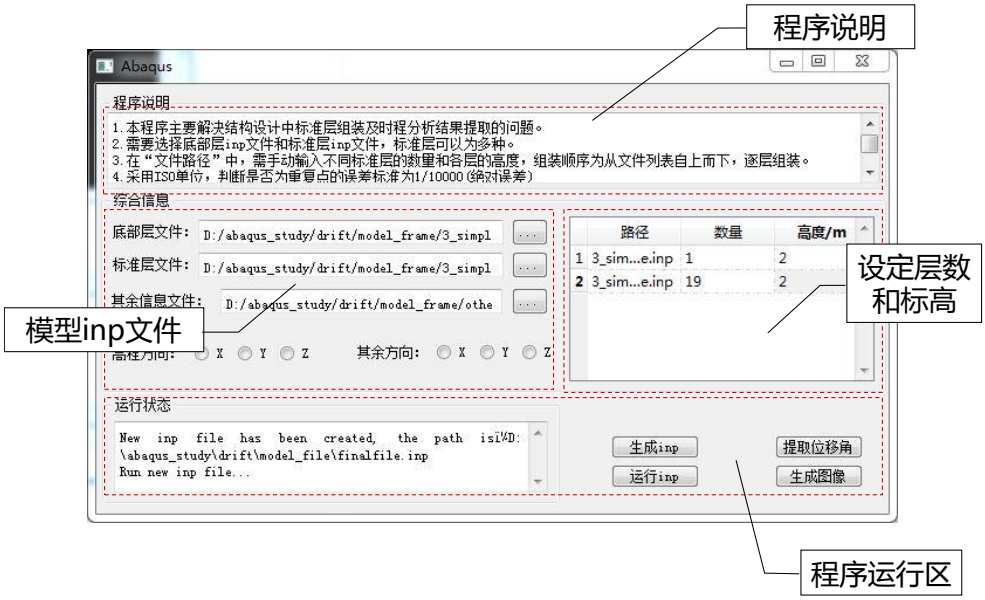


图 7 程序界面

最终实现的程序界面如上图 7，主要分为四个部分：

1) 程序说明部分
2) 输入结构的综合信息部分，包括底部层文件路径，标准层文件路径（可以多个），其余信息文件路径，以及选择高程方向
3) 不同标准层文件对应的数量和高度，其中每输入一种新的标准层，对应的列表会新增一行；同时，组装的顺序是自列表从上往下
4) 运行状态部分，包含运行状态窗口和运行命令按钮

4.3. 程序结果

在编制该软件的过程中，选用的基础模型为平面纤维模型框架，梁、柱的混凝土和钢筋均采用清华大学潘鹏老师开发的 PQ-Fiber 单轴本构模型。选用该基本模型，进行组装，其中基础层为 1 层，上部标准层为 19 层，层高均为 2m，故共 20 层，40m。具体信息如下：

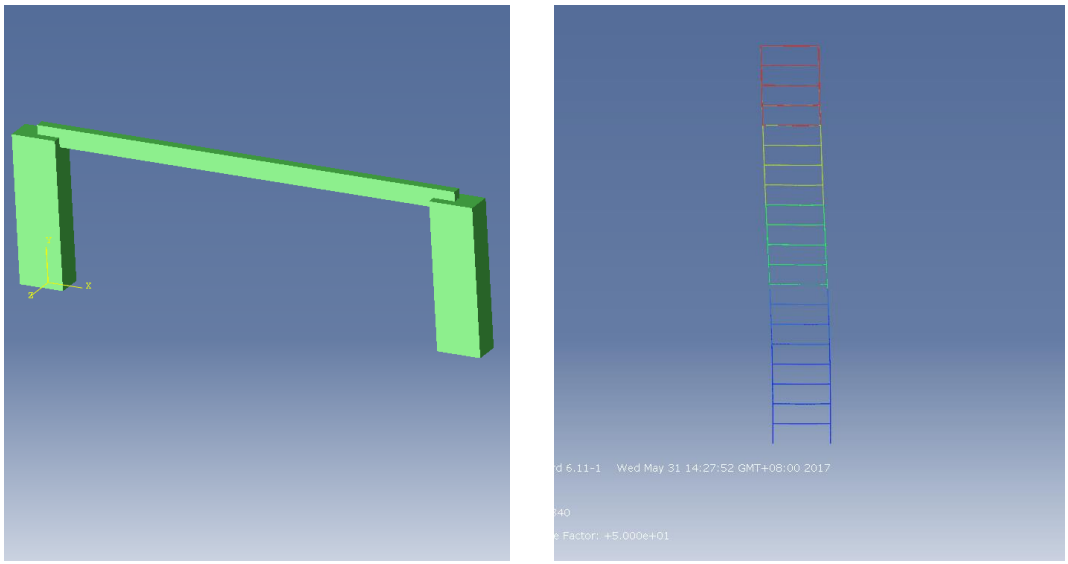


图 8 基础模型&整体模型

选用的地震波为一条人工波 Acc，选取峰值附近的 5~25s 内共 20s 参数；考虑大震作用，峰值加速度取为 220gal，以加速度的方式施加到整体结构上。

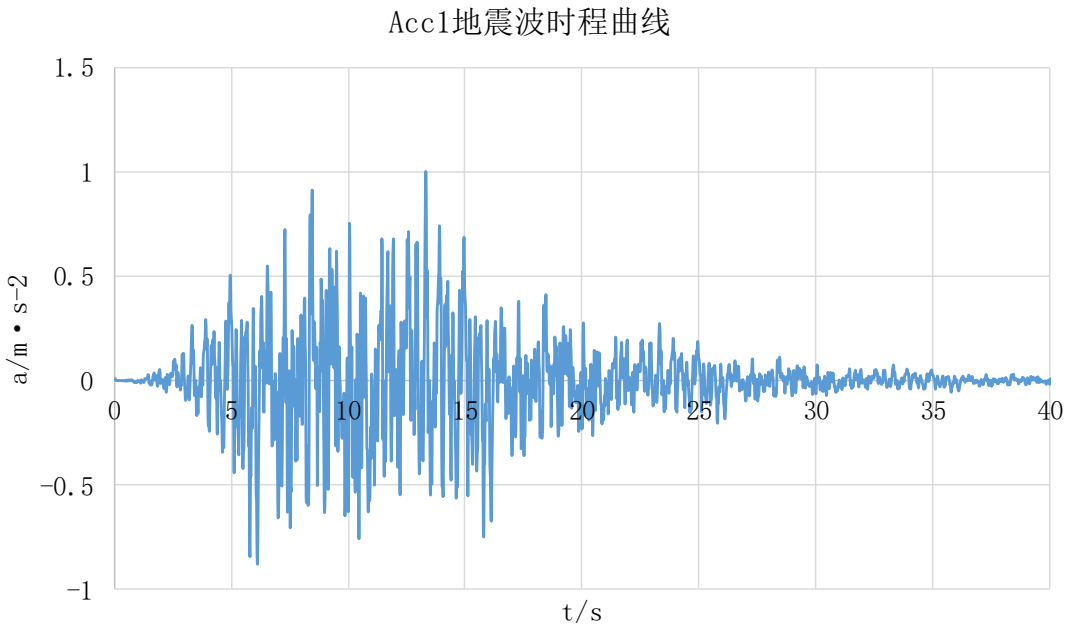


图 8 地震波时程曲线

在组装模型完成之后，进行时程分析，并通过脚本文件的方式，调用 Abaqus 内核，来提取计算结果中的层间位移角数据，记录到 csv 文件中并生成图像，如下图 9 所示。

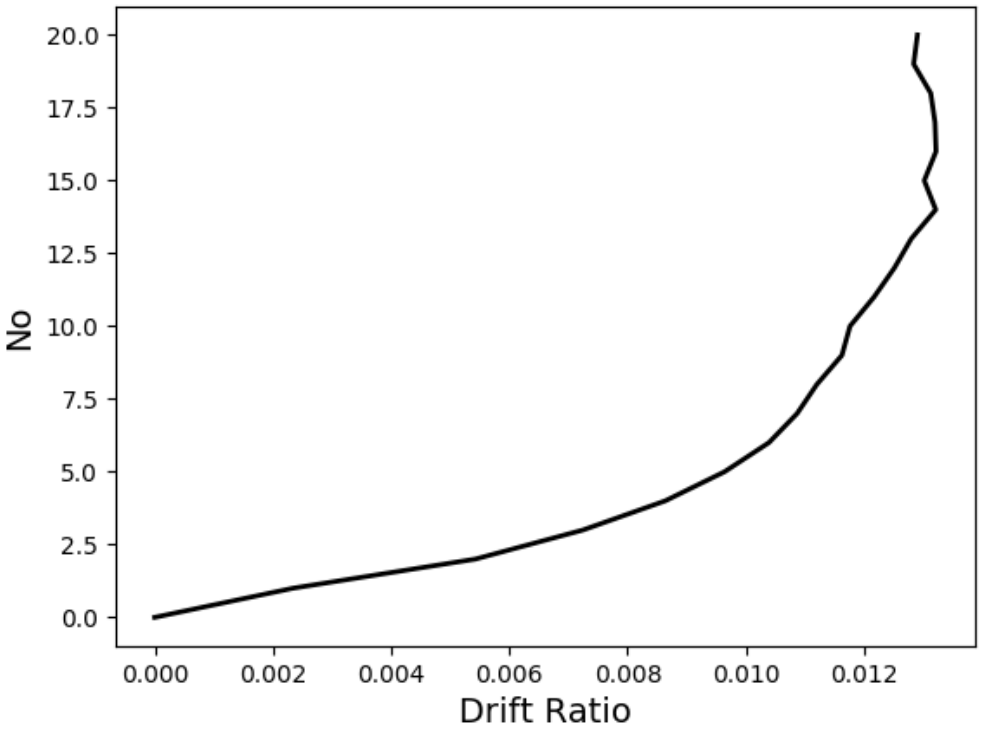


图 9 弹塑性层间位移角曲线

可以看出，弹塑性层间位移角满足规范中要求的钢筋混凝土框架结构$[\theta_p]$=1/50 的要求，因此基本满足设计要求，可以进一步分析其余响应参数。

以上，便完成了一个简单的平面框架结构的组装和大震弹塑性时程分析的过程，简要验证了程序的思路的可行性，基本达成了研究目标。略有一点遗憾的是没有进一步开发出较为便捷有效的优化设计功能，主要原因在于，提供建筑的目标高度后，需要对不同的标准层的组合进行验证，但由于无法进一步建立当不满足层间位移角的要求时，如何调整不同标准层的组装方式的判断标准，这与结构形式和性能密切相关，所以一般来说只能通过遍历来实现，但遍历的计算量过大，达不到简化设计的目的，因此，设计人员可根据每一次的组装的结果和工程经验，进行手动调整，这样会更高效省时。

五、 心得体会

经过本次对于 ABAQUS 二次开发的研究之后，一方面基本完成了研究之初预定的目标，有一些小小的成就感；另一方面，我觉得我收获更多的是对于有限元软件以及对于软件二次开发这个问题的理解。

诚然，对于结构工程师而言，随着计算机技术和有限元分析方法的发展，我们已经基本不会需要进行手算和繁杂的设计成果考查，大部分工作都可以由现有的设计软件、分析软件代劳，这充分体现了人类智慧的发展水平之高，但与此同时也降低了我们对于结构设计本质的理解，而变得更多的依赖于软件提供的参数，成为软件的奴隶。这不仅仅使得我们很难再日常的设计工作当中继续发挥我们设计人员的主观能动性，而且还束缚住了我们的思想，难以拜托软件现有的思路进行创新性的设计工作。而针对这个问题，我觉得软件的二次开发提供了一个很好的机会。它提供了我们在现有设计软件、有限元分析软件的体系化思路之外，另一种可能性，将软件真正便成为我们所用，由个性化的需求出发，服务于设计工作、科研工作等等。

正是基于以上想法，我在学期初选择了这个方向，并结合平时自身的观察和了解选择了对于 ABAQUS 进行二次开发从而引入“标准层”的概念来简化设计工作的思路，做出了一些简单的成果。我在之后的研究生科研工作中，也将把在这次的小研究中学习到的想法和技能充分利用起来，学有所用，学以致用。

六、 对于课程的建议

首先，非常感谢胡老师和林导一学期的悉心教学，从这门课程中，我了解到了很多建筑结构行业 CAE 领域前沿的知识和成果，让我开拓了眼界，受益匪浅。胡老师主导的轻松愉悦的上课交流讨论的氛围也让我印象深刻，当然在这个方面我们学生有些不足，在课堂上不够积极主动。而本着希望这门课程能够越来越好的初衷，我也仔细思考并提出一点粗略的建议，还望胡老师审阅：

1) 由于是小班教学，而且很多的课程主要关注于 CAE 领域前沿的发展，对于有些同学来说可能并不是那么了解。我觉得可以尝试在现有的课程模式下，再新增一个同学们自主选择领域并了解学习后进行课堂展示的环节，这样可能可以更好地加深大家对于相对陌生的问题的了解和学习程度。

最后，再次向胡老师致以谢意，您辛苦了！

附录 A（主程序脚本文件源代码）

```
#-*- coding: UTF-8 -*-
import re
import sys,os,os.path
import shutil
import math
from PyQt4 import QtCore, QtGui,uic
from PyQt4.QtGui import *
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import spline

qtCreatorFile = "Abaqus_3.ui" # Enter file here.

Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)

Up_filename=[]

class MyApp(QtGui.QMainWindow, Ui_MainWindow):
    def __init__(self):
        QtGui.QMainWindow.__init__(self)
        Ui_MainWindow.__init__(self)
        self.setupUi(self)
        self.BottomFile.clicked.connect(self.FindBottomFile)
        self.UpFile.clicked.connect(self.FindUpFile)
        self.OtherFile.clicked.connect(self.FindOtherFile)
        self.Createinp_cmd.clicked.connect(self.Createinp)
        self.Getdrift_cmd.clicked.connect(self.Getdrift)
        self.Runinp_cmd.clicked.connect(self.Runinp)
        self.ShowFig_cmd.clicked.connect(self.ShowFig)
    #-----------------------------
    #定义底部层寻找文件函数
    #-----------------------------
    def FindBottomFile(self):
        global Bottom_filename
        self.PathTable.setRowCount(self.PathTable.rowCount()+1)
        Bottom_filename = QtGui.QFileDialog.getOpenFileName(self, 'Open file', './')
        self.BottomFilePath.setPlainText(str(Bottom_filename))
        Bottom_filename_short = Bottom_filename.split('/')[-1]
        self.PathTable.setItem(0,0,QTableWidgetItem(str(Bottom_filename_short)))
        self.PathTable.setItem(0,1,QTableWidgetItem(str('1')))
    #-----------------------------
    #定义上部标准层寻找文件函数
```

```
    #------------------------------
    def FindUpFile(self):

        self.PathTable.setRowCount(self.PathTable.rowCount()+1)
        Up_filename.append(QtGui.QFileDialog.getOpenFileName(self, 'Open file', './'))
        self.UpFilePath.setPlainText(str(Up_filename[-1]))
        Up_filename_short = str(Up_filename[-1]).split('/')[-1]
        self.PathTable.setItem(self.PathTable.rowCount()-
1,0,QTableWidgetItem(str(Up_filename_short)))

    #------------------------------
    #定义其余信息寻找文件函数
    #------------------------------
    def FindOtherFile(self):
        global Other_filename
        Other_filename = QtGui.QFileDialog.getOpenFileName(self, 'Open file', './')
        self.OtherFilePath.setPlainText(str(Other_filename))

    #------------------------------
    #定义整合 inp 文件函数
    #------------------------------
    def Createinp(self):

        #------------------------------
        #读取底部层 inp 文件
        #------------------------------
        global Bottom_file
        global Bottom_Data
        Bottom_file = open(Bottom_filename)
        Bottom_Data = Bottom_file.readlines()

        #------------------------------
        #读取标准层 inp 文件
        #------------------------------
        Up_file=[]
        Up_Data={}
        Up_No = len(Up_filename)
        for i in range(0,Up_No):
            Up_file.append(open(Up_filename[i]))
            Up_Data[i+1] = Up_file[i].readlines()

        #------------------------------
        #创建整体 inp 文件:finalfile.inp
        #------------------------------
```

```
        global curdir
        curdir = os.getcwd()
        global filepath
        filepath = curdir + "\\" +"model_file"
        if os.path.exists(filepath):
            shutil.rmtree(filepath)
        os.mkdir(filepath)
        Final_file=open(filepath + "\\" + "finalfile.inp",'w')
        shutil.copyfile(curdir + "\\" + 'PQFiberImp_v2.0-std-win64.obj',filepath + "\\" +
'PQFiberImp_v2.0-std-win64.obj')

        #------------------------------
        #读取各类标准层对应的层数,及各层高度
        #------------------------------
        Up_Number=[]
        for i in range(0,Up_No):
            Up_Number.append(int(self.PathTable.item(i+1,1).text()))
            #Final_file.write(str(Up_Number))
        Floor_H = []
        for i in range(0,Up_No+1):
            Floor_H.append(float(self.PathTable.item(i,2).text()))

        #------------------------------
        #读取不同关键词的行数信息
        #------------------------------
        index_Node = {}
        index_Element = {}
        index_Elset_column = {}
        index_Elset_beam = {}
        index_BeamSec_column = {}
        index_BeamSec_beam = {}
        index_EndPart = {}

        index_Node['Bottom'] = Bottom_Data.index('*Node\n')
        index_Element['Bottom'] = Bottom_Data.index('*Element, type=B31\n')
        index_Elset_column['Bottom'] = Bottom_Data.index('*Elset, elset=column\n')
        index_Elset_beam['Bottom'] = Bottom_Data.index('*Elset, elset=beam\n')
        index_EndPart['Bottom'] = Bottom_Data.index('*End Part\n')
        for Bottom_Data_temp in Bottom_Data:
            if '*Beam Section, elset=column,' in Bottom_Data_temp:
                index_BeamSec_column['Bottom'] =
Bottom_Data.index(Bottom_Data_temp)
            if '*Beam Section, elset=beam,' in Bottom_Data_temp:
                index_BeamSec_beam['Bottom'] = Bottom_Data.index(Bottom_Data_temp)
```

```
        for i in range(0,Up_No):
            Up_Data_Beamsec_temp = []
            index_Node['Up-' + str(i+1)] = Up_Data[i+1].index('*Node\n')
            index_Element['Up-' + str(i+1)] = Up_Data[i+1].index('*Element, type=B31\n')
            index_Elset_column['Up-' + str(i+1)] = Up_Data[i+1].index('*Elset, elset=column\n')
            index_Elset_beam['Up-' + str(i+1)] = Up_Data[i+1].index('*Elset, elset=beam\n')
            index_EndPart['Up-' + str(i+1)] = Up_Data[i+1].index('*End Part\n')
            for Up_Data_temp in Up_Data[i+1]:
                if '*Beam Section, elset=column,' in Up_Data_temp:
                    Up_Data_Beamsec_temp = Up_Data[i+1]
                    index_BeamSec_column['Up-' + str(i+1)] = Up_Data_Beamsec_temp.index(Up_Data_temp)
                if '*Beam Section, elset=beam,' in Up_Data_temp:
                    Up_Data_Beamsec_temp = Up_Data[i+1]
                    index_BeamSec_beam['Up-' + str(i+1)] = Up_Data_Beamsec_temp.index(Up_Data_temp)

        #------------------------------
        #读取底部层 Node 信息
        #------------------------------
        Node_info = {}
        Bottom_Data_Node_temp = {}
        for i in range(index_Node['Bottom']+1,index_Element['Bottom']):
            Bottom_Data_apart = []
            Bottom_Data_apart_real = []

            Bottom_Data_apart = Bottom_Data[i].split(',')
            for Bottom_Data_apart_temp in Bottom_Data_apart:

    Bottom_Data_apart_real.append(str(re.findall(r"\d+\.?\d*",Bottom_Data_apart_temp))[2:-2])
            Bottom_Data_Node_temp[Bottom_Data_apart_real[0]] = [Bottom_Data_apart_real[1],Bottom_Data_apart_real[2],Bottom_Data_apart_real[3]]

        Node_info['Bottom']= Bottom_Data_Node_temp

        #------------------------------
        #读取底部层 Element 信息
        #------------------------------
        Element_info = {}
        Bottom_Data_Element_temp = {}
        for i in range(index_Element['Bottom']+1,index_Elset_column['Bottom']):
```

```
            Bottom_Data_apart = []
            Bottom_Data_apart_real = []

            Bottom_Data_apart = Bottom_Data[i].split(',')
            for Bottom_Data_apart_temp in Bottom_Data_apart:

    Bottom_Data_apart_real.append(str(re.findall(r"\d+\.?\d*",Bottom_Data_apart_temp))[2:
-2])
            Bottom_Data_Element_temp[Bottom_Data_apart_real[0]]                    =
[Bottom_Data_apart_real[1],Bottom_Data_apart_real[2]]

        Element_info['Bottom']= Bottom_Data_Element_temp

        #------------------------------
        #读取底部层 Elset 信息
        #------------------------------
        Elset_info = {}
        Bottom_Data_Elset_temp = {}
        Bottom_Data_Elset_temp['column'] = []
        Bottom_Data_Elset_temp['beam'] = []
        for i in range(index_Elset_column['Bottom']+1,index_Elset_beam['Bottom']):
            Bottom_Data_apart = []

            Bottom_Data_apart = Bottom_Data[i].split(',')
            for Bottom_Data_apart_temp in Bottom_Data_apart:

    Bottom_Data_Elset_temp['column'].append(str(re.findall(r"\d+\.?\d*",Bottom_Data_apart
_temp))[2:-2])

        for i in range(index_Elset_beam['Bottom']+1,index_BeamSec_column['Bottom']):
            Bottom_Data_apart = []

            Bottom_Data_apart = Bottom_Data[i].split(',')
            for Bottom_Data_apart_temp in Bottom_Data_apart:

    Bottom_Data_Elset_temp['beam'].append(str(re.findall(r"\d+\.?\d*",Bottom_Data_apart_t
emp))[2:-2])

        Elset_info['Bottom']= Bottom_Data_Elset_temp

        #------------------------------
        #标准层信息读取
        #------------------------------
        for i in range(0,Up_No):
```

```
            Up_Data_temp = Up_Data[i+1]

            #------------------------------
            #读取标准层 Node 信息
            #------------------------------
            Up_Data_Node_temp = {}
            for j in range(index_Node['Up-' + str(i+1)]+1,index_Element['Up-' + str(i+1)]):
                Up_Data_apart = []
                Up_Data_apart_real = []

                Up_Data_apart = Up_Data_temp[j].split(',')
                for Up_Data_apart_temp in Up_Data_apart:

    Up_Data_apart_real.append(str(re.findall(r"\d+\.?\d*",Up_Data_apart_temp))[2:-2])
                    Up_Data_Node_temp[Up_Data_apart_real[0]] =
[Up_Data_apart_real[1],Up_Data_apart_real[2],Up_Data_apart_real[3]]

            Node_info['Up-' + str(i+1)]= Up_Data_Node_temp

            #------------------------------
            #读取标准层 Element 信息
            #------------------------------
            Up_Data_Element_temp = {}
            for j in range(index_Element['Up-' + str(i+1)]+1,index_Elset_column['Up-' +
str(i+1)]):
                Up_Data_apart = []
                Up_Data_apart_real = []

                Up_Data_apart = Up_Data_temp[j].split(',')
                for Up_Data_apart_temp in Up_Data_apart:

    Up_Data_apart_real.append(str(re.findall(r"\d+\.?\d*",Up_Data_apart_temp))[2:-2])
                    Up_Data_Element_temp[Up_Data_apart_real[0]] =
[Up_Data_apart_real[1],Up_Data_apart_real[2]]

            Element_info['Up-' + str(i+1)]= Up_Data_Element_temp

            #------------------------------
            #读取标准层 Elset 信息
            #------------------------------
            Up_Data_Elset_temp = {}
            Up_Data_Elset_temp['column'] = []
            Up_Data_Elset_temp['beam'] = []
```

```
            for j in range(index_Elset_column['Up-' + str(i+1)]+1,index_Elset_beam['Up-' +
str(i+1)]):
                Up_Data_apart = []
                Up_Data_apart = Up_Data_temp[j].split(',')
                for Up_Data_apart_temp in Up_Data_apart:

    Up_Data_Elset_temp['column'].append(str(re.findall(r"\d+\.?\d*",Up_Data_apart_temp))[
2:-2])

            for j in range(index_Elset_beam['Up-' +
str(i+1)]+1,index_BeamSec_column['Up-' + str(i+1)]):
                Up_Data_apart = []
                Up_Data_apart = Up_Data_temp[j].split(',')
                for Up_Data_apart_temp in Up_Data_apart:

    Up_Data_Elset_temp['beam'].append(str(re.findall(r"\d+\.?\d*",Up_Data_apart_temp))[2:
-2])

            Elset_info['Up-' + str(i+1)]= Up_Data_Elset_temp

        #------------------------------
        #整合得到整体 inp 信息
        #------------------------------
        Total_Node = {}
        Total_Element = {}
        Total_Elset = {}
        Total_Nset = {}
        Total_H = {}

        for key,value in Node_info['Bottom'].items():
            Total_Node[int(key)] = [float(value[0]),float(value[1]),float(value[2])]
        for key,value in Element_info['Bottom'].items():
            Total_Element[int(key)] = [int(value[0]),int(value[1])]
        for key,value in Elset_info['Bottom'].items():
            Total_Elset[key] = []
            for v in value:
                Total_Elset[key].append(int(v))
        #------------------------------
        #双层循环，先是对不同的标准层循环，然后对每一种标准层的个数进行循环。逐
层添加时，先添加点，再添加单元，再添加单元组
        #------------------------------
        for i in range(0,Up_No):

            for j in range(0,Up_Number[i]):
```

```
                Node_compare = {}
                for key,value in Node_info['Up-' + str(i+1)].items():
                    value_temp_1 = float(value[0])
                    Floor_No = 0
                    for t in range(0,i):
                        Floor_No = Floor_No + Up_Number[t]
                    if j==0:
                        H_temp=Floor_H[i]
                    else:
                        H_temp=Floor_H[i+1]
                    value_temp_2 = float(value[1])+H_temp*(Floor_No+j+1)
                    value_temp_3 = float(value[2])
                    key_temp = int(key)
                    #------------------------------
                    #判断是否与已有的点重合，重合的话只记录原始点的编号和已有点的编号，若不重合，还需将新的坐标添加到点坐标的字典中
                    #------------------------------
                    flag = 0
                    for k,v in Total_Node.items():
                        if abs((value_temp_1 - v[0]))<0.0001 and abs((value_temp_2 - v[1]))<0.0001 and abs((value_temp_3 - v[2]))<0.0001 :
                            Node_compare[key_temp] = k
                            flag = 1
                    if flag == 0 :
                        Node_No = len(Total_Node)
                        Total_Node[Node_No+1] = [value_temp_1,value_temp_2,value_temp_3]
                        Node_compare[key_temp] = Node_No+1
                #------------------------------
                #添加 element
                #------------------------------
                Element_compare = {}
                for key,value in Element_info['Up-' + str(i+1)].items():
                    value_temp_1 = int(value[0])
                    value_temp_2 = int(value[1])
                    key_temp = int(key)
                    #------------------------------
                    #根据 Node_compare 中的对应关系，把新添加的 element 的点编号更改为新的点编号
                    #------------------------------
                    Element_No = len(Total_Element)
                    Total_Element[Element_No+1] = [Node_compare[value_temp_1],Node_compare[value_temp_2]]
```

```
                    Element_compare[key_temp] = Element_No+1
                #-----------------------------
                #添加 elset
                #-----------------------------
                for key,value in Elset_info['Up-' + str(i+1)].items():
                    if key == 'column':
                        for v in value:
                            Total_Elset['column'].append(Element_compare[int(v)])
                    if key == 'beam':
                        for v in value:
                            Total_Elset['beam'].append(Element_compare[int(v)])

        #-----------------------------
        #将各高程处取一个点，之后建立点集，读取位移数据
        #-----------------------------
        Total_H['Floor-'+'1'] = Floor_H[0]
        Floor_Num = 1
        for l in range(0,Up_No):
            for m in range(0,Up_Number[l]):
                Floor_Num = Floor_Num + 1
                Total_H['Floor-'+str(Floor_Num)] = Total_H['Floor-'+str(Floor_Num-1)] +
Floor_H[l+1]

        for n in range(0,Floor_Num):
            Nset_temp = []
            H_temp = Total_H['Floor-'+str(n+1)]
            for k,v in Total_Node.items():
                if abs(H_temp-v[1])<0.0001:
                    Nset_temp.append(k)
            Total_Nset['Floor-'+str(n+1)] = Nset_temp

        #注释行信息
        Final_file.write('**please check inp carefully!\n')
        #part 信息
        Final_file.write('*Part, name=Part-1\n')
        #Node 信息
        Final_file.write('*Node\n')
        for key,value in Total_Node.items():
            Final_file.write('%g, %g, %g, %g \n'%(key,value[0],value[1],value[2]))
        #Element 信息
        Final_file.write('*Element, type=B31\n')
        for key,value in Total_Element.items():
            Final_file.write('%g, %g, %g \n'%(key,value[0],value[1]))
        #elset 信息
```

```
Final_file.write('*Elset, elset=column\n')
i=1
j=1
for No in Total_Elset['column']:
    Final_file.write('%g'%(No))
    if j < len(Total_Elset['column']):
        Final_file.write(', ')
        j = j + 1
        i = i + 1
        if i >10:
            Final_file.write('\n')
            i = 1
Final_file.write('\n')

Final_file.write('*Elset, elset=beam\n')
i=1
j=1
for No in Total_Elset['beam']:
    Final_file.write('%g'%(No))
    if j < len(Total_Elset['beam']):
        Final_file.write(', ')
        i = i + 1
        j = j + 1
        if i >10:
            Final_file.write('\n')
            i = 1
Final_file.write('\n')
#Nset 信息
Final_file.write('*Nset, nset=Floor-0'+'\n'+' 1, 5'+'\n')
for i in range(0,Floor_Num):
    Final_file.write('*Nset, nset=Floor-'+str(i+1)+'\n')
    temp_1 = 1
    temp_2 = 1
    for node in Total_Nset['Floor-'+str(i+1)]:
        Final_file.write('%g'%(node))
        if temp_1 <len(Total_Nset['Floor-'+str(i+1)]):
            Final_file.write(', ')
            temp_1 = temp_1 + 1
            temp_2 = temp_2 + 1
            if temp_2 >10:
                Final_file.write('\n')
                temp_2 = 1
    Final_file.write('\n')
#----------------------------
```

```
            #写入其他信息，此部分可进一步完善
            #-----------------------------
            Other_file_Data = []

            Other_file=open(Other_filename)
            Other_file_Data=Other_file.readlines()
            for i in range(0,len(Other_file_Data)):
                Final_file.write(Other_file_Data[i])

            #关闭整合后 inp 文件
            Final_file.close()
            self.ProcessShow.append('New inp file has been created, the path is：'+ filepath + "\\" + "finalfile.inp")

        def Runinp(self):
            curdir = os.getcwd()
            filepath = curdir + "\\" +"model_file"
            os.chdir(filepath)
            cmd = 'call abaqus job=finalfile.inp cpus=2 user=PQFiberImp_v2.0-std-win64.obj interactive'
            os.system(cmd)
            self.ProcessShow.append('Run new inp file...')

        def Getdrift(self):
            drift_path = curdir + '\\' + 'drift_py'
            os.chdir(drift_path)
            cmd = 'abaqus cae noGUI=drifts.py' + '\n' + 'pause'
            os.system(cmd)
            self.ProcessShow.append('Get the dirft of floors, the path is：'+drift_path+ '\\' + 'result.csv')

        def ShowFig(self):
            max_drift = []
            max_drift.append(0.0)
            floor_No = []
            floor_No.append(0)
            i=1
            curdir = os.getcwd()
            odbpath = curdir + '\\' +   'result.csv'
            f = open(odbpath,'r')
            ff = f.readlines()
            for fff in ff:
                f_temp = re.sub('\D','',fff)
                max_drift.append(f_temp)
```

```
                floor_No.append(i)
                i = i + 1

            N = np.array(floor_No)
            D = np.array(max_drift)
            #D = D.astype('S32')
            #Nnew = np.linspace(N.min(),N.max(),300)
            #Nnew = Nnew.astype('S32')pyplot
            #Dnew = spline(N,D,Nnew)

            plt.close('all')
            #plt.plot(Dnew,Nnew,c="k",lw=2,label="Ratio of Story Drifts Curve")
            plt.plot(D,N,c="k",lw=2,label="Ratio of Story Drifts Curve")
            plt.xlabel("Drift Ratio",fontsize=14)
            plt.ylabel("No",fontsize=14)
            plt.show()

if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    window = MyApp()
    window.show()
sys.exit(app.exec_())
```

附录 B（提取层间位移角脚本文件源代码）

```
from odbAccess import *
from abaqusConstants import *
import os

curdir = os.getcwd()
odbpath = os.path.dirname(curdir) + '\\' + 'model_file' + '\\' + 'finalfile.odb'
f = open(curdir + '\\' + 'result.csv','w')

myodb = openOdb(odbpath)
mysteps = myodb.steps

mystep = mysteps['earthquake']
myframes = mystep.frames
myframeNo = len(myframes)
```

```
nodeset = myodb.rootAssembly.instances['PART-1-1'].nodeSets.keys()

floorNO = 0
for nset_temp in nodeset:
    if nset_temp[:6] == 'FLOOR-':
        floorNO = floorNO + 1

max_drift=[]
for h in range(0,floorNO-1):
    Upfloor = 'FLOOR-' + str(h+1)
    Downfloor = 'FLOOR-' + str(h)
    x = []
    h_1 = myodb.rootAssembly.instances['PART-1-1'].nodeSets[Upfloor].nodes[-1].coordinates[-2]
    h_2 = myodb.rootAssembly.instances['PART-1-1'].nodeSets[Downfloor].nodes[-1].coordinates[-2]
    h = abs(h_1 - h_2)

    Up_nset = myodb.rootAssembly.instances['PART-1-1'].nodeSets[Upfloor]
    Down_nset = myodb.rootAssembly.instances['PART-1-1'].nodeSets[Downfloor]
    Up_NodeNum = len(Up_nset.nodes)
    Down_NodeNum = len(Down_nset.nodes)

    for frame_No in range(0,myframeNo):
        frames = myframes[frame_No]
        dis = frames.fieldOutputs['U']
        Up_dis = dis.getSubset(region = Up_nset)
        Down_dis = dis.getSubset(region = Down_nset)
        Up_nodevalue = Up_dis.values
        Down_nodevalue = Down_dis.values
        u1 = []
        u2 = []
        for t1 in Up_nodevalue:
            u1.append(t1.data[0])
        for t2 in Down_nodevalue:
            u2.append(t2.data[0])
        x.append(sum(u1)/Up_NodeNum-sum(u2)/Down_NodeNum)
    maxvalue=max(max(x),abs(min(x)))
    max_drift.append(maxvalue/h)

floor_No=[]
for i in range(0,floorNO-1):
    f.write('FLOOR-%d&%d-DRIFT,%g \n' %(i,i+1,max_drift[i]))
    floor_No.append(i+1)
```

```
myodb.close()
f.close()
```