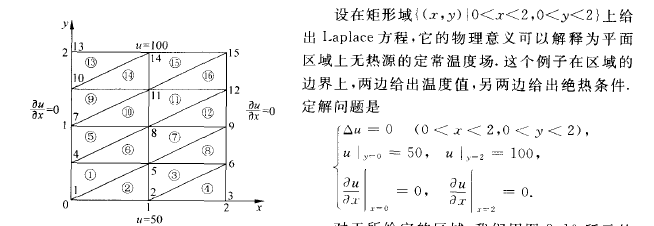
# 说 明

## ****1、问题描述****

本项目是为解决以下常微分方程而创建的：



本项目使用有限元方法，分别对三角形剖分和矩形剖分进行处理，其中三角形剖分借用easymesh软件剖分的网格。

本文件内包含：

* 包含3套不同疏密的easymesh三角形网格剖分文件夹
* easymesh生成的文件说明书: EasyMesh 说明书.docx
* 基本版的解决方案： magic\_tool.py
* 进阶版的解决方案： more\_powerful\_magic\_tool.py

进阶版解决方案用户可以选择引入非齐次项、其他本质边界条件。

编程环境：

编译器：python 3.5

函数库：numpy 1.14.2

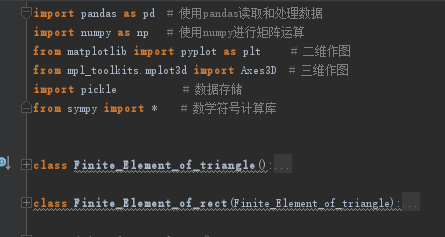
pandas 0.18.1

matplotlib 1.5.3

sympy 1.0

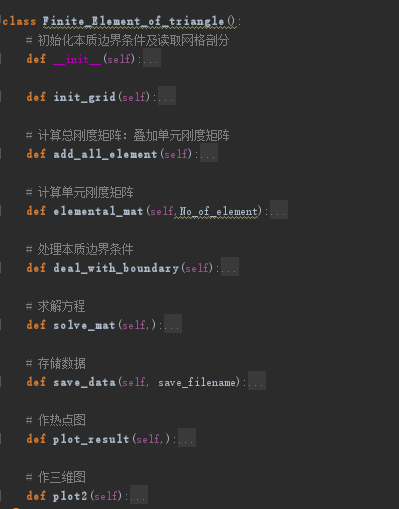
## 2、magic\_tool.py说明

该文件用于解决1中问题。结构如下：

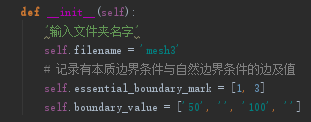


该文件包含Finite\_Element\_of\_triangle及Finite\_Element\_of\_rect类，分别用于解决三角形剖分和矩形剖分。

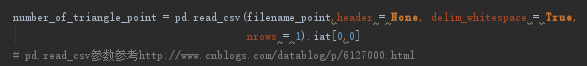
**Finite\_Element\_of\_triangle类结构如下图所示：**

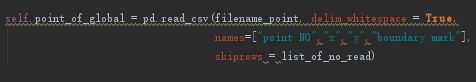


在 \_\_init\_\_中，用户可更改需要读取的文件夹名称或修改本质边界条件：



init\_\_grid方法用于从文件夹中读取网格数据。在本题中，边界信息没有应用就暂不读取，但留下了读取时需要用到的代码。





pd.read\_csv用于读取数据文件，并创建pandas库中DataFrame类型文件。需要使用到的参数：

header ：创建列标题

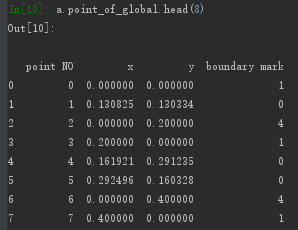
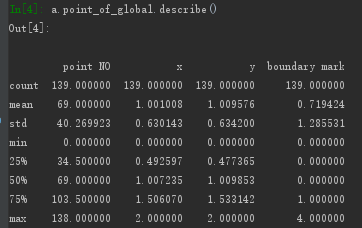
delim\_whitespace：设置是否将空格作为分隔符，

nrows ：需要读取的行

names：自定义列标题

skiprows：跳过的行数

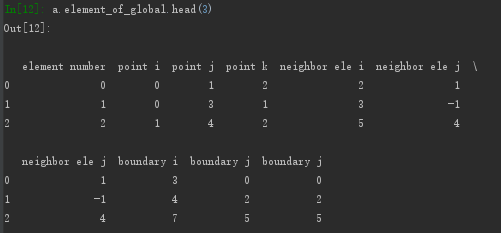
DataFrame类型可调用iat方法读取数据中特定行和特定列的数据，或调用iloc调用特定一行或一列数据。也可使用describe、head、tail等方法查看数据：



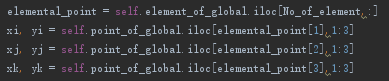
add\_all\_element方法用于将单元刚度矩阵叠加到总刚度矩阵:

elemental\_mat方法用于生成单元刚度矩阵:

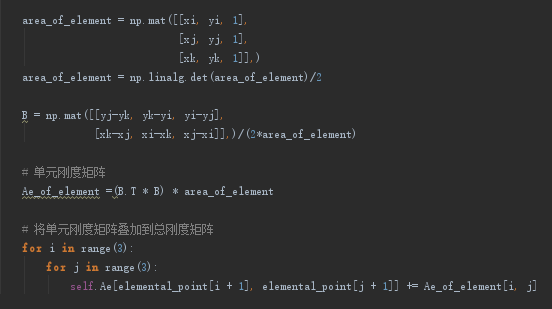
首先需要读取单元信息，单元信息包含在element\_of\_global中：



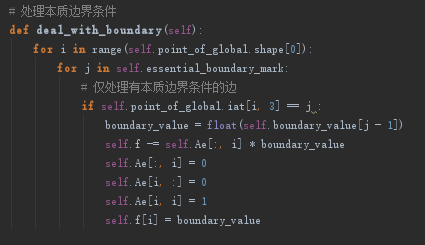
使用iloc方法读取第No\_of\_element个单元中的信息，并记录单元的三个节点坐标。



由于此时常微分问题，故后续单元刚度矩阵也较为简单：

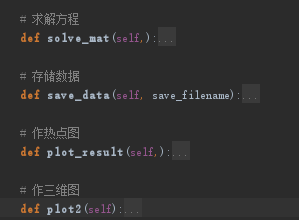


deal\_with\_boundary方法用于处理本质边界条件：

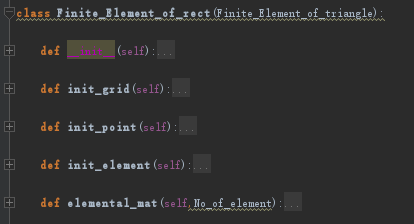


essential\_boundary\_mark变量用于记录给出的本质边界条件的边标志，boundary\_value记录对应的值，两者皆已在\_\_init\_\_中初始化。

总刚度矩阵和总载荷矩阵都已经生成，后续是求解及数据后处理。

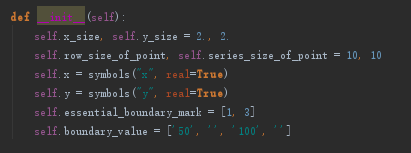


**Finite\_Element\_of\_rect类结构如下图所示：**



该类继承了Finite\_Element\_of\_triangle类，但矩形剖分的基函数与三角形剖分不同，且不能使用easymesh的三角形网格剖分，于是改写和添加了一些方法。

\_\_init\_\_中，初始化了x、y的大小，以及节点行列数，由于矩形元计算单元刚度矩阵涉及积分，对x、y做了符号化处理：

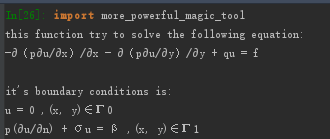


init\_grid、init\_point与init\_element则对网格节点和网格单元进行了划分。

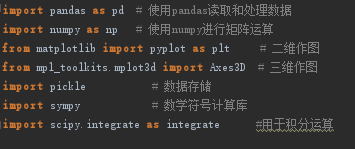
elemental\_mat生成矩形元对应的单元刚度矩阵并叠加到总刚度矩阵当中。

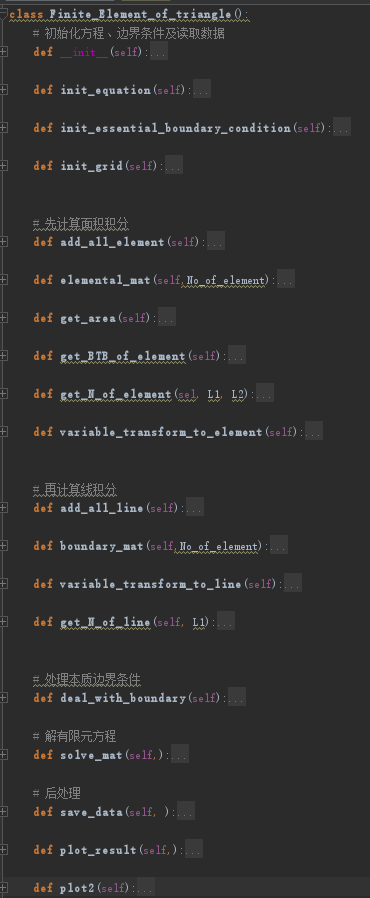
## 3、more\_powerful\_magic\_tool.py说明

有限元方法中，三角形剖分相对矩形剖分更具有普适性，本文件舍弃了矩形剖分对应的Finite\_Element\_of\_rect类，并在magic\_tool基础上对Finite\_Element\_of\_triangle的改进，用户可以输入常微分方程及本质边界条件：



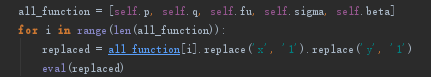
需要载入的模块：





同magic\_tool.py中一样，\_\_init\_\_方法初始化要读取的文件夹以及本质边界条件，另外，此时还可以修改方程中的各个函数。这三种功能分别对应了三个方法。

其中init\_equation用于初始化方程，输入回车可以使用默认设置值。值得注意的是，如果由用户输入值，还需要判断输入是否合法



目前判断输入是否合法的模块还比较初级，只能输入多项式，后续还有待改进。

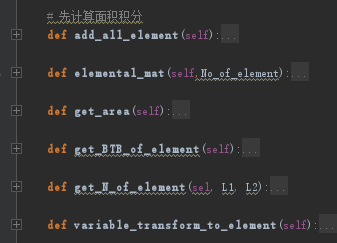
init\_essential\_boundary\_condition方法用于初始化本质边界条件，输入回车可以使用默认设置值。如果由用户输入值，同样需要判断输入是否合法：



此模块中默认了4个边界，后续可以根据读取的easymesh文件判断边界个数，再分别赋值。

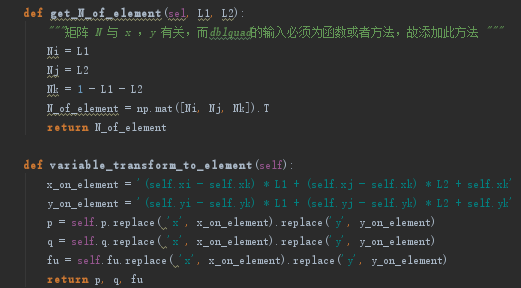
用户输入的值都使用字符串保存。

接下来是面积积分的计算：



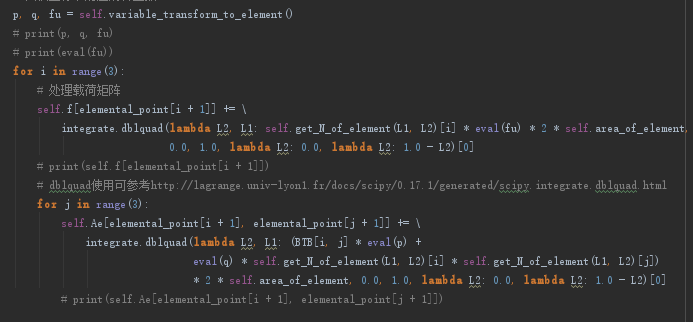
需要注意的是，此时的常微分方程已经不是齐次方程，面积积分的计算必须用到积分。一种方法是直接积分，另一种方法是使用Gauss求积公式。由于python计算积分有比较方便的模块，此处使用的是直接积分。

在积分之前，需要做一些处理，使用面积积分计算积分会方便一些。



get\_N\_of\_element方法是获得对应的面积坐标，variable\_transform\_to\_element是将各个函数转变为面积坐标表示。

此时的elemental\_mat方法只需更改单元刚度矩阵和单元载荷矩阵的计算即可：



integrate模块中的dblquad函数用于二重积分计算，可参考http://lagrange.univ-lyon1.fr/docs/scipy/0.17.1/generated/scipy.integrate.dblquad.html

线积分的计算类似，只是需要注意此时的节点编号不再保证是逆时针，需要对get\_area的返回值做绝对值处理。

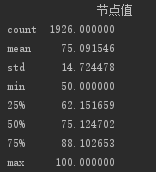
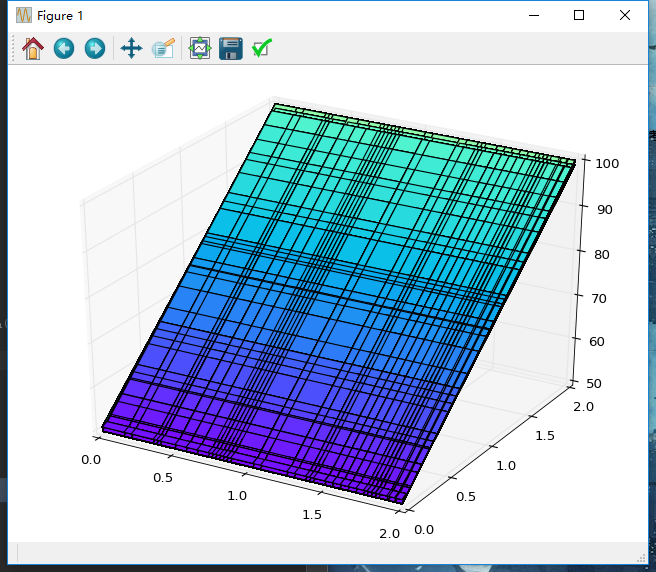
## 4、不足与后续扩展

不足之处：除了初始化方程过程可输入方程不能为复杂方程外，本项目还存在不能剖分三角形网格的缺点。故Finite\_Element\_of\_triangle类还只能处理例题中的模型，后续读取其他网格剖分时，可以通过修改init\_essential\_boundary\_condition方法将本质边界条件与边界标志一一对应。另外，python用于数学计算的计算速度比较比较慢。

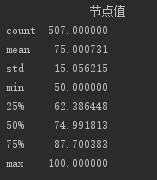
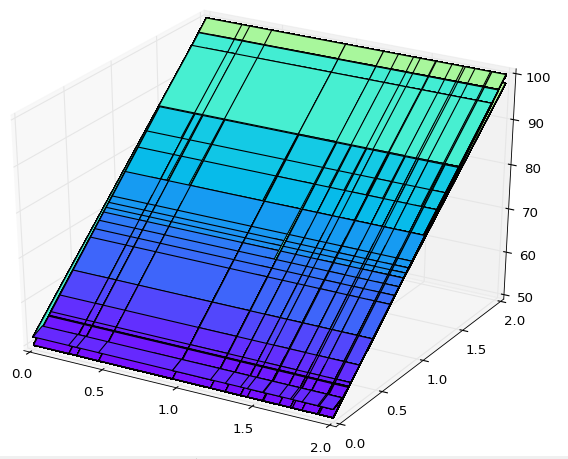
后续扩展：可以考虑将easymesh整合到该项目中，以完成有限元方法的封闭处理。

## 5、结果验证与误差分析

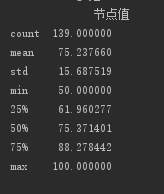
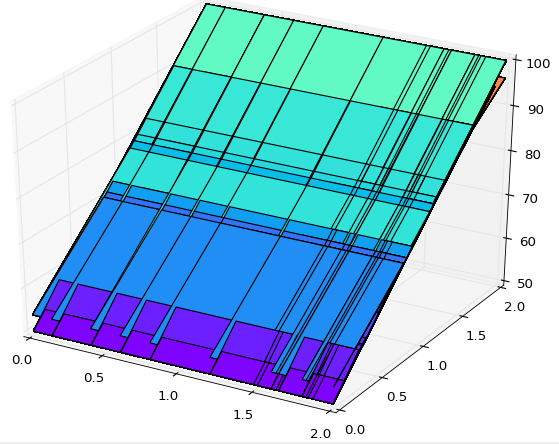
使用mesh1文件夹中网格剖分的结果：



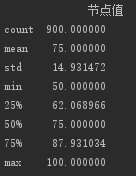
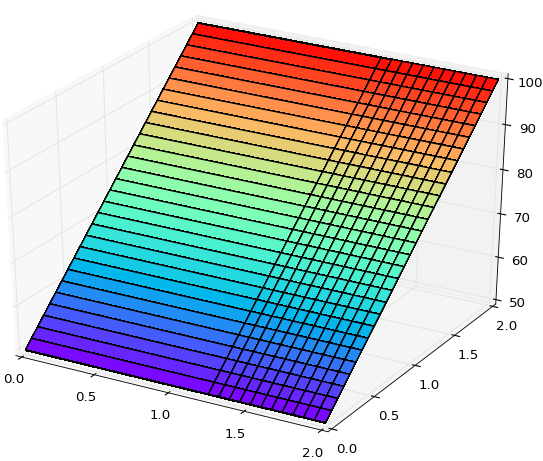
使用mesh2文件夹中网格剖分的结果：



使用mesh3文件夹中网格剖分的结果：



行列数分别为30、30时的矩形剖分结果：



事实上，原问题有精确解：u = 50 + 25 \*y。可以看到，四个结果与精确解符合得很好。

对于三角形剖分而言，节点值存在一定的误差，这部分误差是由于用剖分网格来与模型存在误差导致的。可以看到，当剖分节点越多，网格越密集时，误差也越小。当引入非齐次项时，也会由于计算机存储位数有限的原因导致积分计算过程中存在误差。

虽然本模型中矩形元的误差较小，但不建议使用该中网格剖分。因为在一些比较复杂的网格中，矩形元近似会导致较大误差。

## 6、学习心得与体会

有限元方法课程告一段落。但是对有限元方法的学习却不会停歇。有限元方法与差分方法是解决常微分方程的二大数值方法。相对于差分方法，有限元方法的逻辑更为严密，适用性更强，但计算量较大。

在本课程的学习过程中，一方面，感受到了数学严谨的逻辑思维带来的乐趣。例如变分法引理中，由于要求了u是连续的，于是由连续函数的保号性很自然地得出结论。另一方面，体会到了将数学理论知识转化为实际解决问题的程序时所带来的创造乐趣。在这个过程中，不仅对理论知识的掌握有很高的要求，还需要不断查文档来进行编程实行，给人以无涯学海中遨游的乐趣。