Gmsh 初步学习
并行计算第一次作业

郑灵超

2016 年 10 月 21 日

目录

1 Gmsh 软件介绍

1.1 软件简介

Gmsh 是一款内嵌了 CAD 和后处理的三维有限元网格生成器。它能提供一种快速、轻便、方便使用且有良好的可视化效果的网格生成工具。

Gmsh 主要有四个模块：几何体，网格，求解器和后处理, 在本报告中，我们将着重关心 Gmsh 中几何体的生成、网格的剖分，以及利用 Gmsh 生成的网格结合 AFEPack 软件包进行有限元问题的求解。

本报告只是作者学习 Gmsh 软件的一些经验，纰漏之处在所难免，读者可以通过下载 Gmsh 官方帮助文档 (http://gmsh.info/doc/texinfo/gmsh.pdf) 获取更多信息。

1.2 软件安装

Gmsh 的安装途径十分简单

- 直接在 ubuntu 源中安装：终端输入 sudo apt-get install gmsh 即可。
- 在 Gmsh 官网 (http://gmsh.info) 下载软件包，解压后可以直接运行 bin 文件夹中的 gmsh 文件。

1.3 软件使用方式

- 直接在终端输入 gmsh, 打开图形界面，之后可以生成新的文件或者在图形界面中打开指定的文件。
- 在命令行输入 gmsh file，打开图形界面并载入制定文件。
- 通过命令行模式执行，例如
  >gmsh t1.geo -2
  将会对指定文件 t1.geo 进行 2 维的网格剖分。

关于软件使用形式的

由于实际问题中结构体（点，线，面，体）的数目较多，在使用中我们建议自己撰写或用生成 geo 文件，再用 gmsh 打开并操作。

1.4 Gmsh 的优势与不足

1.4.1 Gmsh 的优势

- 能够较快地描述几何体。
- 能够确定几何体的参数。

- 可以比较容易地生成一些简单的 1D,2D,3D 有限元网格，并与 CAD 软件有着较好的适应性。
- 可以精细地控制网格尺寸。
- 可以通过现有几何体的变换得到新的几何体。
- 自带了有限元的求解器。

1.4.2 Gmsh 的不足

- 用于描述几何体的方法——BRep approach 在处理大规模模型时会不再方便。
- Gmsh 生成的网格较为有限，只能剖分适合有限元方法的网格。
- Gmsh 使用的文件，例如.geo 文件的语言稍显落后，例如循环系统较为简陋，没有自定义的宏和局部变量。
- Gmsh 没有撤销操作，将会带来一些不便。

2 几何结构的构造

几何结构是 Gmsh 中最基本的元素，之后的网格剖分和方程计算都将基于构造好的几何体。Gmsh 不仅可以让用户自己从点开始生成几何体，也可以基于已有的几何体经过变换和拉伸得到。

下面我们介绍一些最基本的几何结构的生成方式以及它们的作用和变换形式。

2.1 Point(点)

- `Point(n)={x,y,z,lc};`
  这个命令可以在坐标 (x, y, z) 处插入一个编号为 n 的点；lc 是特征尺度，表示该点所属网格的尺度，用于控制网格的疏密程度，lc 可以省略。

2.2 Line(线)

- `Line(n)={n1,n2};`
  可以生成一条从第 n_1 号点到 n_2 号点的线段，线段编号为 n。

- `Circle(n)={n1,n2,n3 }`;
  可以生成一条弧度小于 π 的圆弧，圆弧的圆心编号为 n_2，起点编号为 n_1，终点编号为 n_3。

2.3 Loop(环)

Loop 是 Gmsh 的几何结构相对于传统的欧式几何结构的不同之处，分为 Line Loop 和 Surface Loop。其中 Line Loop 可以用来定义面，而 Surface Loop 可以用于定义体，此处我们先介绍 Line Loop 的产生。

```
Line Loop(n)={expression list};
```

用于产生 line loop. 其中 n 为 line loop 的编号，右端的 experssion list 包含所有构成这个 line loop 的线。构成 line loop 的所有 line 必须是封闭且统一方向的。

2.4 Surface(面)

- `Plane Surface(n)={expression list};`
  用于生成一个平面区域，其中 n 为区域编号，右端的 expression 包含了若干个 Line Loop，其中第一个 Line Loop 是产生该区域的外边界，其余 Line Loop 为该区域的内边界（洞）。

- `Ruled Surface(n)={expression list} <In Sphere{expression}>;`
  用于生成一个直纹曲面，其中 n 和 expression list 与 Plane Surface 中的相同，分别

为编号和 Line Loop; <> 中为可选的参数，表示该曲面限定在一个球面上并指定球心点编号。

- `Surface Loop(n)={expression list};`
  用于生成 surface loop, 以构造体。其中 n 为 surface loop 的编号，右边为构成这个 surface loop 的所有面的编号，右边参数要保证 surface loop 是封闭的且自洽的（所有面法向指向同一个区域）。

- `Compound Surface(n)={expression list};`
  用于生成一个复合平面。复合平面由若干平面组成，这些平面在网格剖分的时候被视为同一个平面（同一个网格可能跨过平面边界）。

2.5 Volume(体)

- `Volume(n)={expression list};`
  可以创建一个体。与创建面类似，expression list 包含若干个 Surface Loop，其中第一个为该几何体的外边界，其余几个为内边界 (孔)。

- `Compound Volume(n)={expression list};`
  可以创建一个组合的几何体。与组合面类似，组合体在进行网格剖分的时候网格可以跨越边界。

2.6 Extrusions

通过 Extrude 命令，我们可以从点生成线，从线生成面，从面生成几何体。

- `Extrude {x,y,z} {extrude list};`
  使得将 extrude list 中的元素沿着 (x, y, z) 方向移动对应距离，并将初始位置和最终位置对应连接。
  例如如果作用于点，生成的就是线；作用于线，生成的就是面；作用于面，生成的就是几何体。

- `Extrude{{x1,y1,z1},{x2,y2,z2},theta}{extrude list};`
  将 Extrude list 中所有元素进行旋转操作，并根据轨迹生成新的几何结构。其中 (x_1, x_2, x_3) 给出了旋转轴的方向，(x_2, y_2, z_2) 给出了旋转轴上的一个点，θ 给出了旋转角度 (弧度)。

- `Extrude {{x,y,z},{x1,y1,z1},{x2,y2,z2},theta}{extrude list};`
  将 Extrude list 中的元素进行平移加旋转操作，平移和旋转的参数分别由 (x, y, z) 和 $(x_1, y_1, z_1), (x_2, y_2, z_2), \theta$ 给出，规则与前文一致。

2.7 Transformations

Transformation 作用于几何体将不再生成轨迹，而只是生成结果，默认将原数据覆盖，也可以生成一个复制品 (利用 Duplicata 命令)。

- `Dilate {{x,y,z},alpha}{transform list};`
  将 transform list 所有元素进行放缩，其中 (x, y, z) 给出了位似变换的方向，而 α 给出放缩的比例。
- `Rotate {{x1,y1,z1},{x2,y2,z2},theta}{transform list};`
  将 transform list 所有元素进行旋转变换。其中 $x_1, y_1, z_1, x_2, y_2, z_2, \theta$ 确定了旋转的参数，形式与 Extrude 中的旋转相同。
- `Symmetry {a,b,c,d} { transform list };`
  将 transform list 中所有元素进行对称操作。其中 a, b, c, d 给出了对称平面方程的四个系数。
- `Translate {x,y,z} {transform list};`
  将 transform list 中所有元素沿着向量 (x, y, z) 平移。
- `Boundary { transform list};`
  返回 transform list 中所有元素的边界。
- `CombinedBoundary { transform list };`
  返回 transform list 所有元素总的边界。

2.8 Physical entity(物理实体)

由于完全使用 Element entities 将带来一些不便:

- 网格元素不能被复制。
- 网格元素的节点编号完全由他们的“父”结构决定，而且难以修改。
- 隶属于不同元素结构体的一些元素不能被同一种数学物理的形式表出，如狄利克雷边界条件。

基于以上原因，在实际应用中需要引入一些物理实体。

```
Physical Point(n) = { expression list };
```

其中 n 为物理实体的编号，右端的 expression list 包括所有将被囊括到这个物理点的元素点 (Element Point) 的编号。

3 网格生成

3.1 二维网格

3.1.1 二维三角形网格

对于 Surface，Gmsh 默认将其剖分为三角形网格，因此二维无结构三角形网格只需要按照默认值进行，即终端输入

```
gmsh -2 test.geo
```

就能将 test.geo 文件进行剖分，默认输出为 test.msh。

3.1.2 一个简单的例子

这里我们给出一个简单的网格剖分的例子：

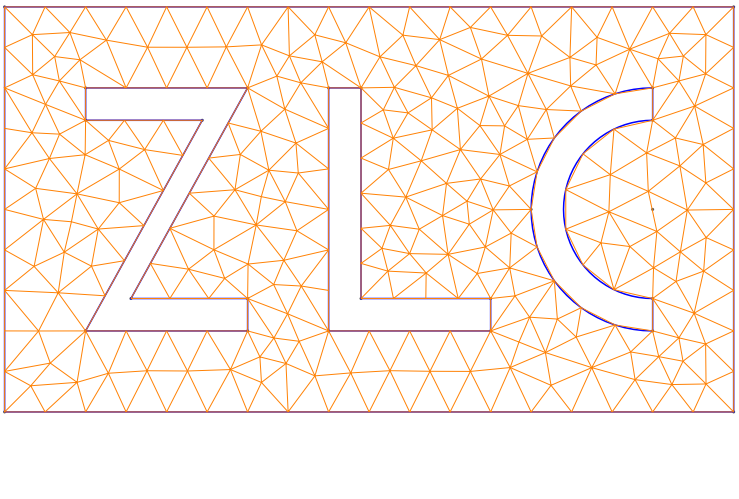


图 1: 简单的例子

以下是实现这个例子的 geo 文件代码，体现了 Gmsh 的基本操作。

```
lc = 0.5;
h = 0.4;
//外边界
Point(1) = {0,0,0,lc};
Point(2) = {9,0,0,lc};
Point(3) = {9,5,0,lc};
Point(4) = {0,5,0,lc};
```

```
Line(1) = {1,2};
Line(2) = {2,3};
Line(3) = {3,4};
Line(4) = {4,1};

Line Loop(5) = {1,2,3,4};

//Z
Point(101) = {1,1,0,lc};
Point(102) = {3-1.4*h,4-h,0,lc};
Point(103) = {1,4-h,0,lc};
Point(104) = {1,4,0,lc};
Point(105) = {3,4,0,lc};
Point(106) = {1+1.4*h,1+h,0,lc};
Point(107) = {3,1+h,0,lc};
Point(108) = {3,1,0,lc};

Line(101)= {101,102};
Line(102)= {102,103};
Line(103)= {103,104};
Line(104)= {104,105};
Line(105)= {105,106};
Line(106)= {106,107};
Line(107)= {107,108};
Line(108)= {108,101};

Line Loop(109) ={101,102,103,104,105,106,107,108};

//L

Point(201) = {4,1,0,lc};
Point(202) = {4,4,0,lc};
Point(203) = {4+h,4,0,lc};
Point(204) = {4+h,1+h,0,lc};
Point(205) = {6,1+h,0,lc};
Point(206) = {6,1,0,lc};

Line(201)={201,202};
Line(202)={202,203};
Line(203)={203,204};
Line(204)={204,205};
Line(205)={205,206};
Line(206)={206,201};

Line Loop(207) = {201,202,203,204,205,206};
```

```
//c

Point(301) = {8,1,0,lc};
Point(302) = {8,4,0,lc};
Point(303) = {8,4-h,0,lc};
Point(304) = {8,1+h,0,lc};
Point(305) = {8,2.5,0,lc};
Point(401) = {6.5,2.5,0,lc};

Circle(301) = {302,305,301};
Line(302) = {302,303};
Circle(303) = {303,305,304};
Line(304) = {304,301};

Line Loop(306) = {-301,302,303,304};

Plane Surface(6) = {5,109,207,306};
```

zlc.geo

3.1.3 网格加密

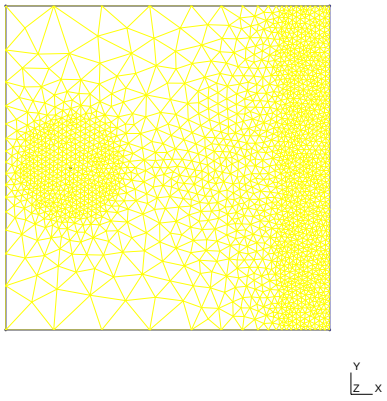
通过 RefineMesh 命令，可以通过将当前的每个网格进行分割，从而完成网格加密。

3.1.4 网格密度设置

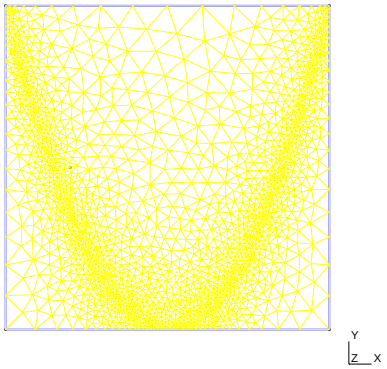
一个简单的设置密度的形式是设置点和曲线的特征尺度 Mesh.CharacteristicLengthFromPionts 和 Mesh.CharacteristicLengthFromCurvature 。

此外通过 Field 命令，可以比较精细地设置网格密度。

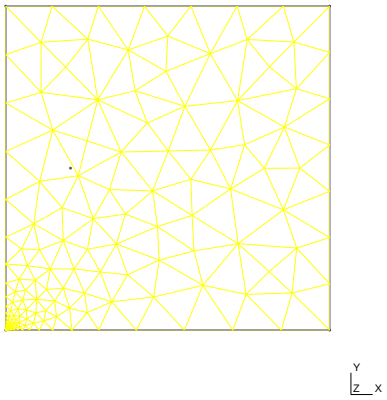
- Threshold 可以设置网格根据和一个几何结构的距离来形成不同密度,其中 Attractor 可以设置网格根据和一点或者一条线的距离来产生不同密度。
- Box 可以设置网格在一个区域内部和外部的疏密程度的不同。
- MathEval 可以设置网格密度为一个函数。
- Min 可以设置网格密度为几个 Field 所设置的最小值。



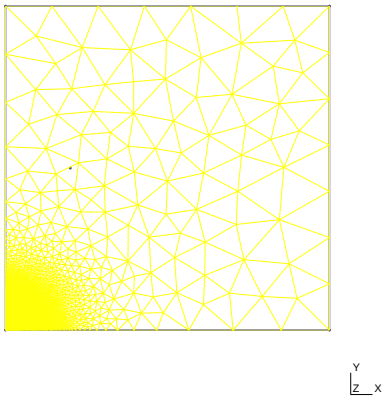
(a) 网格在一个点和一条线附近稠密



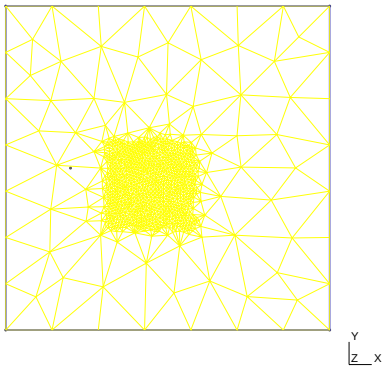
(b) 网格在抛物线附近稠密



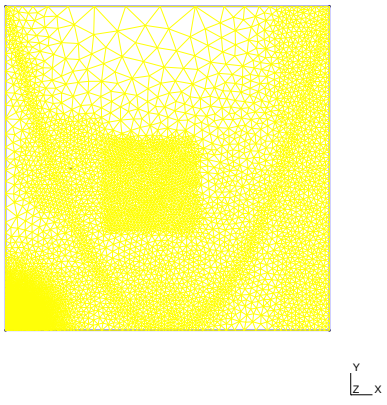
(c) 网格在端点附近稠密



(d) 网格在端点附近加密



(e) 网格在区域内稠密



(f) 网格在若干处同时稠密

上述例子可以通过以下代码实现：

```
// Let's create a simple rectangular geometry
lc = .15;
Point(1) = {0.0,0.0,0,lc}; Point(2) = {1,0.0,0,lc};
Point(3) = {1,1,0,lc};
 Point(4) = {0,1,0,lc};
Point(5) = {0.2,.5,0,lc};
Line(1) = {1,2}; Line(2) = {2,3}; Line(3) = {3,4}; Line(4) = {4,1};
Line Loop(5) = {1,2,3,4}; Plane Surface(6) = {5};


Field[1] = Attractor;
Field[1].NodesList = {5};
Field[1].NNodesByEdge = 100;
Field[1].EdgesList = {2};


Field[2] = Threshold;
Field[2].IField = 1;
Field[2].LcMin = lc / 10;
Field[2].LcMax = lc;
Field[2].DistMin = 0.15;
Field[2].DistMax = 0.5;


Field[3] = MathEval;
Field[3].F = "abs(4*(x-0.5)*(x-0.5)-y)*0.1+0.01";



Field[4] = Attractor;
Field[4].NodesList = {1};
Field[5] = MathEval;
Field[5].F = Sprintf("F4^3 + %g", lc / 100);
Field[6] = Box;
Field[6].VIn = lc / 15;
Field[6].VOut = lc;
Field[6].XMin = 0.3;
Field[6].XMax = 0.6;
Field[6].YMin = 0.3;
Field[6].YMax = 0.6;
Field[7] = Min;
Field[7].FieldsList = {2, 3, 5, 6};
Background Field = 2;
```

field.geo

3.1.5 二维四边形网格

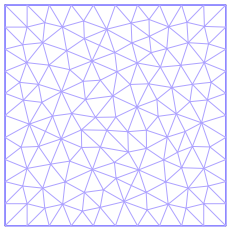
Gmsh 提供了一个 Recombine 函数，二维情形下将三角形网格重新整合为四边形网格。

```
Recombine Surface{expression list};
```

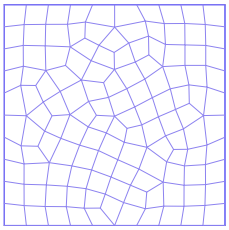
其中 expression list 给出了所有需要整合的 Surface 的编号，或者采用

```
Recombine Surface '*';
```

来一次性整合所有 Surface。



(g) 三角形网格



(h) 四边形网格

3.1.6 二维结构四边形网格

要生成二维结构四边形网格，可以采用 Extrude 来完成。

```
Extrude { expression list} {extrude list layers}
```

其中 expression list 给出了所有要被 Extrude 的几何体。而 extrude list layers 可以加入一些参数来决定生成网格的层数以及是否进行整合。

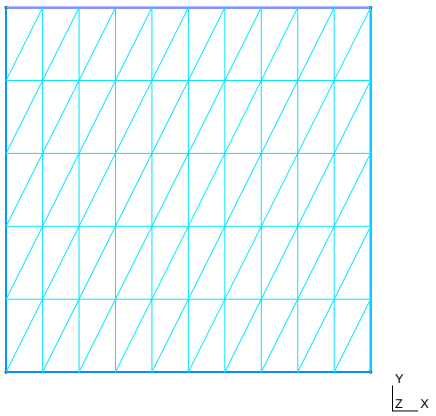
例如

```
Extrude { 0,0.1,0 } {  Line{1};Layers{5}; };
```

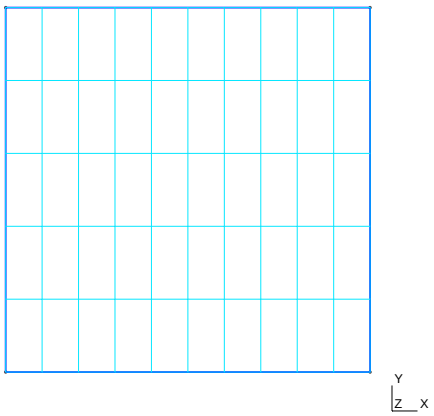
可以生成左图，其中 Layers{5} 表示在移动方向（此处为数值方向）上生成 5 个层次。

```
Extrude { 0,0.1,0 } {  Line{1};Layers{5}; Recombine; };
```

可以生成右图，其中 Recombine 表示要进行整合，整合结果为矩形网格。



(i) 结构三角形网格



(j) 结构四边形网格

3.2 三维网格

3.2.1 三维四面体网格

在对区域进行三维的网格剖分时，Gmsh 默认按照无结构四面体网格剖分，因此只需要

```
gmsh test.geo -3
```

就可以对 test.geo 进行三维的四面体网格剖分。

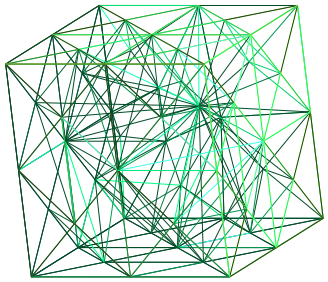
3.2.2 三维结构网格

对于三维区域，若要生成除了四面体之外的网格，需要采用结构网格的形式，即采用 `Extrude` 命令。以下我们给出一个简单的生成长方体网格的代码。

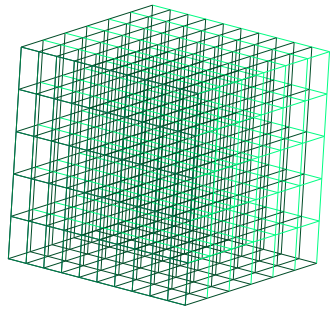
```
//定义初始元素，两个点和一条线。
lc = 1e-2;
h = 0.1;
Point(1) = {0, 0, 0, lc};
Point(2) = {h, 0, 0, lc};

Line(1) = {1,2};
//生成一个平面，按照长方形剖分。
out1[] = Extrude { 0 , h , 0 } {
    Line{1}; Layers{5};
    Recombine;
};

//生成长方体
out2[] = Extrude { 0 , 0, h } {
```

```
    Surface{out1[1]}; Layers{ 6 };
    //将已经生成出来带有长方形网格的平面out1[1]来进行extrude，这样
    //生成的三维网格才会随之产生长方体。
    //如果用三角形网格的Surface进行这步操作，会产生三棱柱网格。
    Recombine;
    };
//定义物理实体
Physical Point(1) = {1,2};
Physical Line(1) = {1};
Physical Surface(1) = {out1[1]};
Physical Volume(1) = {out2[1]};
```

cuboid.geo



(k) 三维四面体网格



(l) 三维长方体网格

3.3 与 Easymesh 的比较

相比于 AFEPack 中使用的 easymesh 软件，Gmsh 是一个成熟的网格剖分系统，在网格剖分部分，二者有以下一些差异之处：

- Gmsh 生成几何体的方式较为简便，可以通过一些基本元素的 Extrude 来生成。而 Easymesh 只能通过逐点描述线段来生成。
- Easymesh 只能处理多边形边界的区域，Gmsh 可以处理维带圆弧或者其余二次曲线的区域。例如在之前给出的有“ZLC”字样的区域，、就无法通过 Easymesh 进行剖分，只能先人为地用多边形去逼近这块区域，再交给 easymesh 剖分。
- Gmsh 可以比较容易地设置网格在某个特定区域内稠密。
- 对于二维问题，Easymesh 只能生成无结构三角形网格，Gmsh 可以生成无结构的三角形、四边形以及结构三角形、四边形网格。
- Gmsh 能够处理三维问题。

4 网格输出文件的结构分析

通过在图形界面下的 Save mesh，或是直接在终端下 `gmsh %.geo -D` ，可以将网格进行剖分并存入文件%.msh。以下我们分析 msh 文件的格式:

```
$MeshFormat
2.2 0 8  //版本号 文件形式 浮点数据占用内存大小
         //这三个数字一般不会变化。
$EndMeshFormat
$Nodes
13  //节点数目
1 0 0 0 //编号和x,y,z坐标
2 1 0 0
......
$EndNodes
$Elements
28  //结构数目，这里的结构可以是点、线、面、体。
1 15 2 1 1 1
// 结构编号 结构性质 tag的数目 若干个tag 所包含的顶点编号
2 15 2 1 2 2
3 15 2 1 3 3
4 15 2 1 4 4
5 1 2 1 1 1 5
6 1 2 1 1 5 2
......
25 2 2 1 6 1 5 12
26 2 2 1 6 3 11 6
27 2 2 1 6 3 7 11
28 2 2 1 6 4 8 10
$EndElements
```

sample.msh

需要注意的是这里的结构性质参数，它的意义如下:

结构性质参数	结构类型
1	2 个节点的线段
2	3 个节点的三角形
3	4 个节点的四边形
4	4 个节点的四面体
5	8 个节点的六面体
6	6 个节点的三棱柱
7	5 个节点的四棱锥
15	1 个节点的点

表 1: 结构类型

我们这里不关心 tag 的意义。

以这份省略版的 sample.msh 为例，它描述了一个一共只有 13 个节点的网格。这个结构体一共有 28 个结构，其中 1-4 号为点（结构性质参数为 15），之后为边（结构性质参数为 1），再之后为三角形（结构性质参数为 2）。因此这是一个三角形网格，且每个三角形网格的顶点都在对应行的末尾给出 (位于 tag 之后)。

5 有限元问题的求解

5.1 AFEPack 对 Gmsh 的接口

在 AFEPack 头文件中，有一个 Mesh 类，是 AFEPack 可以直接处理的网格形式，这种形式的数据无法由 gmsh 的网格文件.msh 文件直接得到。从而 AFEPack 为 Gmsh 写了一个接口类 GmshMesh [1]，用于读入.msh 文件并转化为 Mesh 类的形式。

AFEPack 中给出了六面体、三棱柱等几何体的模板单元，但目前尚未给出能够正确处理 Gmsh 中对应形状网格的函数。目前能够在 AFEPack 中使用的网格形式只有二维的三角形、四边形网格和三维的四面体网格。

5.2 一个简单的例子

我们这里采用 AFEPack 来进行有限元问题的求解。事实上，AFEPack 在读取网格时已经为 Gmsh 写了接口，这里我们使用 AFEPack 读取一个简单的 Gmsh 网格，并求解一个简单的泊松方程。

我们选取的区域是一个单位正方形减去中间的一个圆形区域，这是一个 Easymesh 无法处理的区域形状。具体方程和真实解为：

$$-\Delta u = 5\pi^2 \sin(\pi x)\sin(2\pi y)$$

$$u = \sin(\pi x)\sin(2\pi y)$$

边界条件采用狄利克雷边界条件。

利用 AFEPack 计算的结果显示如下：

[1] AFEPack 源代码这部分有一些小问题，经过唐凤阳师兄修复之后可以使用

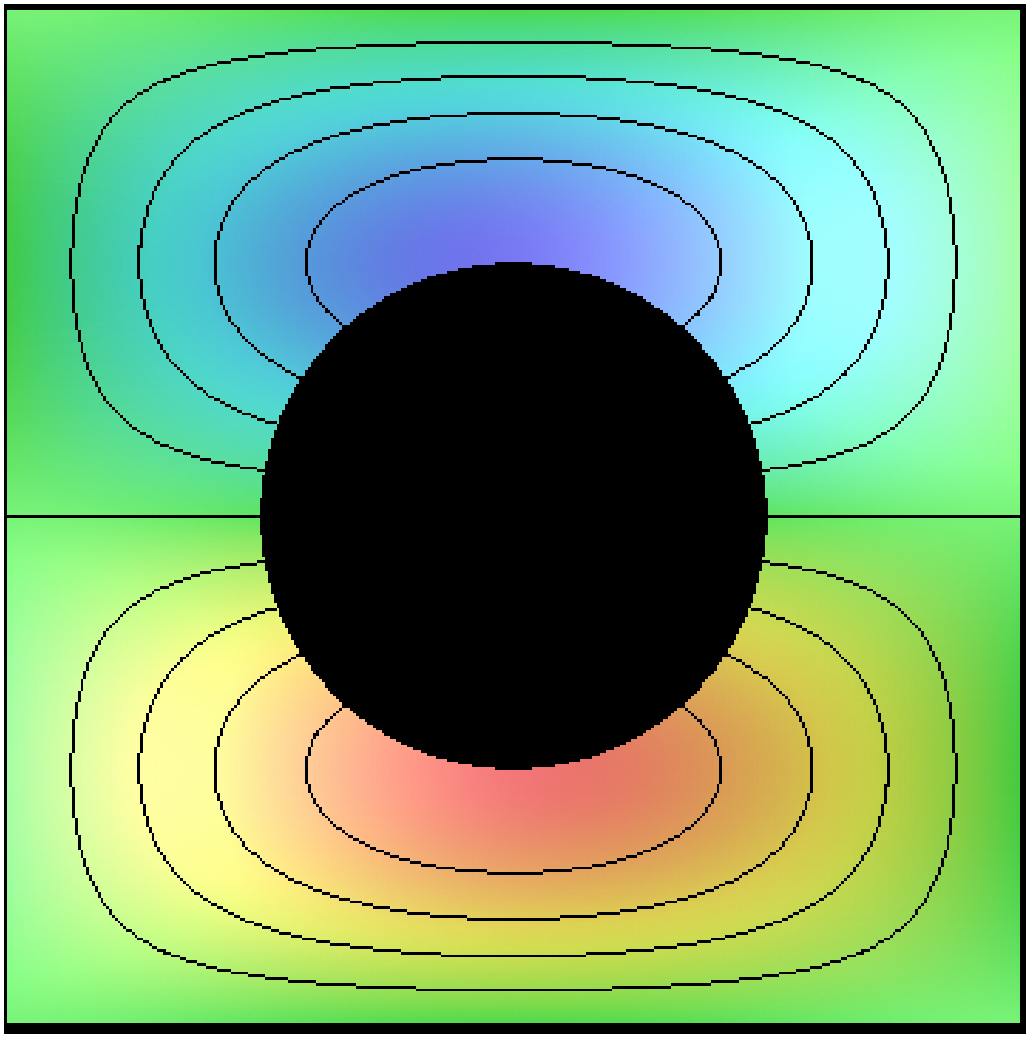


图 2: AFEPack 解泊松方程

其中与真解比较，L2 误差为 0.00013，可以认为 AFEPack 正确解决了这个问题。

通过这个简单的例子，我们搭建了从 Gmsh 剖分网格，到 AFEPack 解决有限元问题的桥梁，使 AFEPack 摆脱了对 Easymesh 的依赖，从而可以处理一些更复杂的区域问题和构建更适用的网格。

6 总结

本次上次作业，我们学习了网格剖分软件 Gmsh, 这是一款网格剖分的专业软件。相比 Easymesh,Gmsh 的适用范围更大，生成几何体的方式更简单，而且能够处理三维的几何结构以及曲线形状的几何体，还能剖分出三角形，四边形，四面体网格和各种结构网格。

此外，我们还利用了 AFEPack 软件包以及它对 Gmsh 的接口，求解了一个简单的二维区域的泊松方程，取得了不错的结果。目前 AFEPack 可以用于求解二维的 Gmsh 网格，以及三维的 Gmsh 生成的四面体网格。其他形状的网格暂时还没有支持的接口。

此次上机作业只记录了一些 Gmsh 的基本用法，还有待更进一步的学习。Gmsh 官方提供的帮助文档十分详细，下载地址为 http://gmsh.info/doc/texinfo/gmsh.pdf.