基于表面三角网格的四面体剖分

杨义军（035456） 龚洁晖（035453） 王文珂（035452）
yang-yj03@mails.tsinghua.edu.cn

摘 要 四面体剖分可以表达模型的拓扑特性，简化模型的分析。对模型特别是工业模型进行四面体剖分是有限元分析的一个重要步骤。本文详细描述了一种四面体剖分生成算法。首先对 Gems™生成的模型用 Ansys™进行表面三角剖分；对 Ansys™处理过的表面三角网格进行三角形的空球检测，对不满足空球特性的三角面片进行必要的边变换；在此基础上，根据晶体网格(FCC) 加点规则在模型内部加入内部点；最后算法采用波前法和 Delaunay 准则生成模型的四面体剖分。本文最后给出了部分剖分结果和中间结果，对算法中出现的一些问题进行了分析。

关键词 晶体网格 八叉树 Delaunay 剖分 波前法

1 问题背景

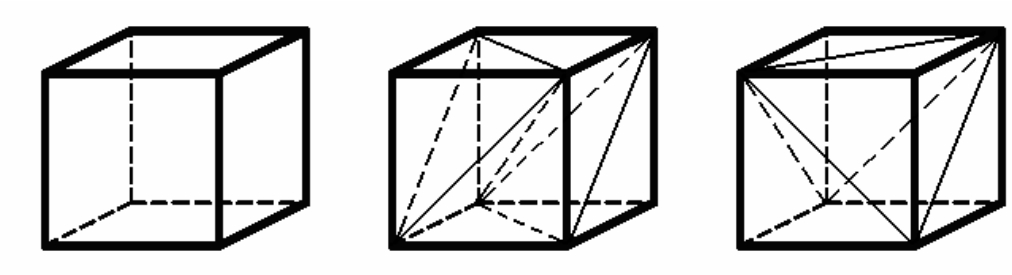


图 1（1） 正方体 （2）正方体一种四面体剖分 （3）另一种四面体剖分

模型的拓扑特性可以用四面体剖分很好的表示，所以四面体剖分是有限元分析等许多工程应用的基础方法。一般模型的四面体不是唯一的，如图 1，即使对一个正方体进行四面体剖分，剖分结果也不唯一。一个模型在没有加入内部点的情况下是不是存在四面体剖分是 NP-Hard 问题。对于图 2 中的多面体，没加入附加点的情况下不存在四面体剖分。但是每一个多面体都能在加入内部点（最坏情况 O(n)）的条件下进行四面体剖分。一个好的四面体剖分应满足以下要求：

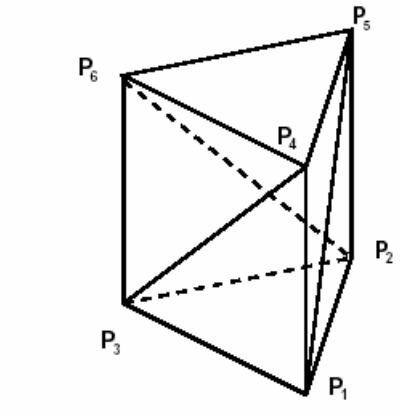


图 2 不可四面体剖分多面体

合法性：　四面体不能相交，在边界上的结点均应作为剖分的结点，不可丢弃。

相容性：　四面体必须落在待分区域内部，且所有四面体的并集等于待分区域。

逼近精确性：待分区域的顶点（包括特殊点）必须是单元的结点，待分区域的边界（包括特殊边及面）被单元边界所逼近。

良好的形状：单元最佳形状是正多边形或正多面体，单元之间过渡应相对平稳。

自适应性：　四面体剖分的密度应符合待分区域的物理特性，在几何尖角处、应力温度等变化大处较密集，其他部位较稀疏。

采用几何方法的四面体剖分技术主要有几何分解法、结点连元法。在产生结点的同时也确定结点间拓扑关系的方法称为几何分解法，常用的有两种：递归法和迭代法。几何分解法的关键技术在于新结点的生成方法，最大优点是在离散物体时考虑网格单元的形状和大小，所生成的网格单元形状和分布均较好；最大缺点是自动化程度低，不利于复杂件网格生成。

结点连元法是先生成结点，然后连接结点构成单元。最常用的是 DT 法和 AFM 法。DT 法的基本原理是利用 Voronoi 图和 Delaunay 三角剖分的对偶性质，连接相邻 Voronoi 多边形的内核点构成 Delaunay 三角剖分。DT 法的最大优点是遵循 Delaunay 三角剖分“最小角最大”和“空球”准则，同时满足全局和局部最优；但直接的 Delaunay 三角剖分只适用于凸域，不适用于非凸域。AFM 法的基本原理是假设区域的有向离散外边界集和前沿边界集已经确定，按某种条件沿区域边界向区域内部削切四面体，同时不断更新前沿边界集，直到空集。AFM 法的关键技术有两个：一是区域的边界离散和内结点的合理生成。二是由当前面选择内部点生成新的四面体的规则。AFM 法最大优点是不仅在区域内部而且在区域边界所生成的网格单元形状均优良，网格生成全自动。

本文实现了一种基于内部点添加和波前法的四面体剖分算法。算法利用了晶体网格的加点规则和 Delaunay 准则，生成的四面体较好地保持了原来多面体的边界特性。算法的时间复杂性为 $O(n^2)$。

2 算法描述

本文首先对表面的三角形进行调整，使得每一个三角形的空球中不包含面上的其他点；接着对面调整后的多面体根据晶体网格规则插入内部点；最后对整个多面体和内部点用波前法和 Delaunay 准则进行四面体剖分。

2.1 空间的八叉树划分

三角形调整时，为了加速三角形外接球包含点的检测以及内部点的添加，本文首先对多面体进行了空间的八叉树划分。八叉树节点分成三类：

八叉树外部节点：完全位于多边体的外部，不包含多面体上的点，标记为 o;

八叉树边界节点：和多面体相交，一部分在多面体的外部，标记为 b;

八叉树内部节点：完全位于多面体的内部，表记为 i;

对多面体建立八叉树时，八叉树的根结点设置成多面体的包围盒。同时把多面体的所有顶点的索引添加到根节点数组中，标记根结点为 b；然后从根节点开始进行递归剖分，如果此节点的递归剖分深度大于一个上限，则不进行剖分，否则：

（1）如果此节点标记为 b, 则将此节点剖分成 8 个子节点，把位于此节点中所有多面体顶点分到 8 个子节点中。如果子节点完全位于多面体内部则标记为 i；如果子节点位于多面体外部则标记为 o；如果部分在多面体外部，部分在多面体内部则标记为 b。

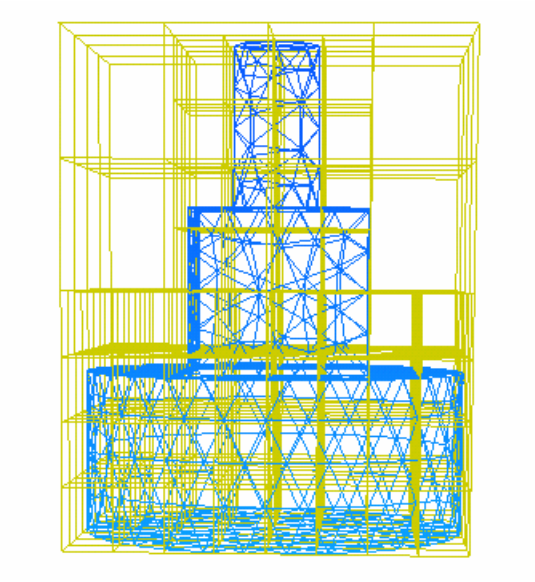


图 3 八叉树剖分后的结果图

（2） 如果此节点标记为 o，则对此节点不进行八叉数剖分。

（3） 如果此节点标记为 i，则把此节点剖分成 8 个子节点，把位于此节点中所有多面体顶点分到 8 个子节点中，所有的子节点标记为 i。

算法的伪代码表示如下：

```
function divide()
begin
    if(此节点深度>规定最大深度)
        return;
    switch(节点标记为)
    begin
        case ‘b’:
                生成八个子节点，把此节点包含的顶点分到八个子节点中，给每个子节点加标记，建立子节点和此节点的关联;
                break；
        case’o’:
              return;
        case’i’ :
                生成八个子节点，把此节点包含的顶点分到八个子节点中,把每个子节点标记为 I, 建立子节点和此节点的关联。
    end
    调用八个子节点每个子节点的 divide 函数。
End
```

图 3 给出了一个多面体进行以上八叉树剖分后生成的结果。

2.2 点在多面体内外的判断

进行空间的八叉树划分以及内部点的添加，经常需要对点在多面体内外进行判断。算法如下，对多面体的每一个面和整个体计算包围盒。

（1）如果一个点在整个体的包围盒的外部则标记为 o；

（2）此点位于整个体包围盒面上，则要判断是否位于多边形一个三角面片内部，如果是则标记为 f,如果位于多面体的一条棱上，则标记为 e,如果和多面体和一个顶点重合则标记为 v。否则标记为 o。

（3）此点位于整个体包围盒内部，则从此点发出一条射线，对这条射线和多面体每一个面的包围盒进行求交判断，如果此射线和一个面的包围盒相交，则对此射线和面进行实际的判交。

统计所有的交点数目，如果是奇数，则此点位于多面体的内部，如果为偶数，此点位于多面体的外部。

2.3 三角形的调整

由于任一多面体不一定存在四面体剖分，所以为了对获得的模型能够进行四面体剖分，本算法首先对面三角形进行空球检测，使得每一个表面的三角形所在的最小球中不包含其它面上点。算法首先计算三角形的外界圆的圆心和半径。△abc 外界圆的可以通过计算线 ab 的垂直平分面和 bc 的垂直平分面，相交于直线 e，e 和平面 abc 的交点即为△abc 的外接圆的圆心，外接圆的半径为外接圆的圆心和三角形任一顶点的距离。三角形外接球的球心和半径与外界圆的圆心和半径是一样的。如果在三角形的外接球中包含模型上其他点，则对三角形和相邻三角形进行对角线交换测试，如果两个三角形位于同一个平面内，且交换对角线后生成的三角形“最小角最大”意义下形状最好。则对此三角形和相邻三角形进行对角线交换。如图 4（a）红色三角形的外界球中包含了模型中的其它点，则对此三角形和相邻的三角形进行了对角线的交换，交换后的结果如图 4（b）,可以发现交换对角线后的两个三角形的外接球中不包含原来位于非法三角形外接球中的点。

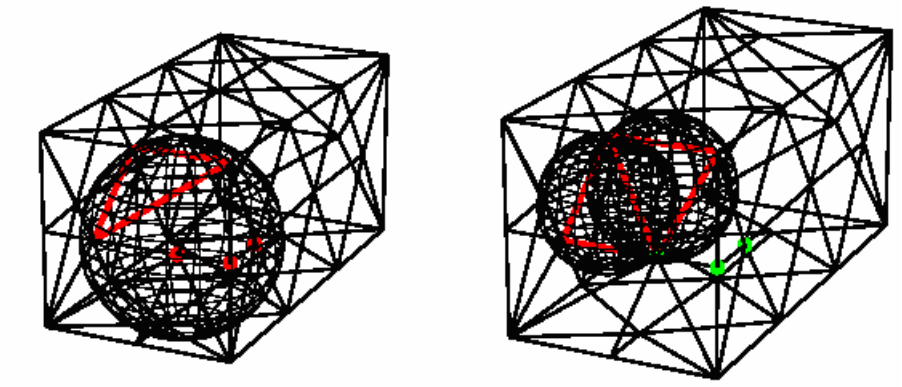


图 4（a）对角线交换前　　（b）对角线交换后

如果非法三角形不能进行对角线的交换，则要对非法三角形的最长边进行中点剖分，同时和此三角形共最长边的另一个三角形也要进行相应的剖分。如图 5（a）最靠近我们的两个三角形都不符合外接球空球条件，每一个三角形的外接球中都包含另外的一个点，我们对这两个三角形进行最长边的进行剖分后结果图如图 5(b)。

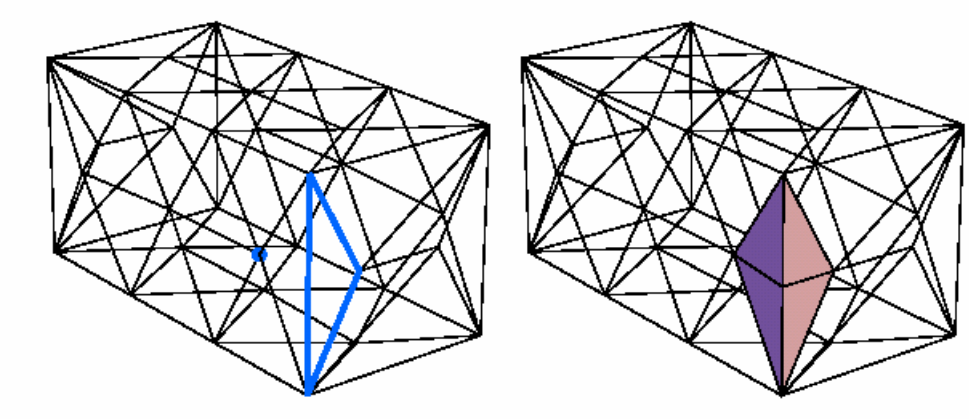


图 5（a）三角形剖分前　　(b)三角形剖分后

对图 6(a)所示多面体进行面的三角形调整后如图 6(b)所示。

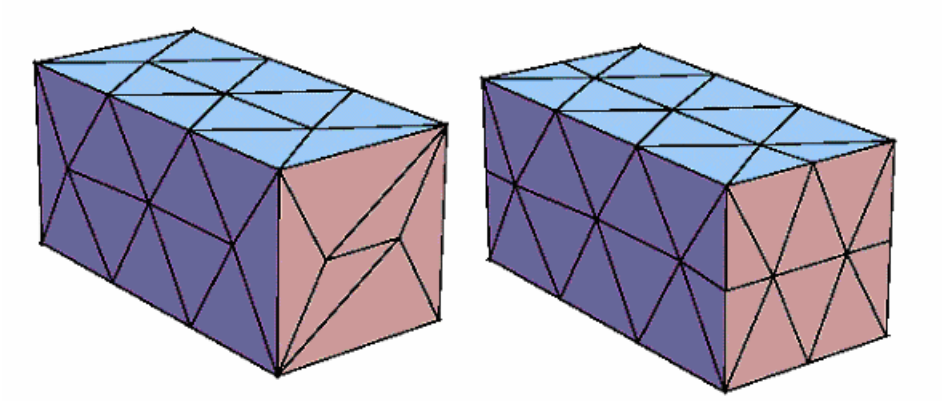


图 6（a）初始三角网格　　(b)调整后三角网格

2.4 内部点的添加

本文对多面体进行八叉树剖分后，将八叉树的每一个叶结点的作为加入内部点的基本单元。存在两种加点规则：面心立方结构(FCC)和体心立方结构（BCC）。分别如图 7 所示。

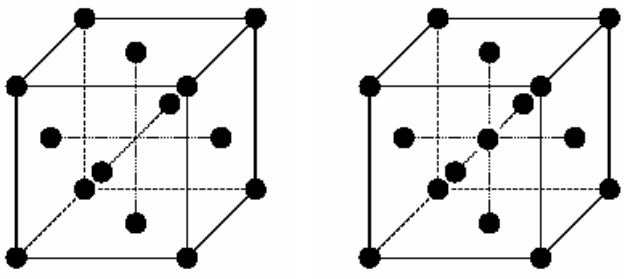


图 7（a）FCC　　(b)BCC

根据统计，面心立方的加点规则生成的四面体质量要好于体心立方。所以我们对每个八叉树叶结点按照面心立方的规则加点，去掉位于多面体外部的节点。

对一个 1*1 的立方体进行面心加点后的剖面图如下：

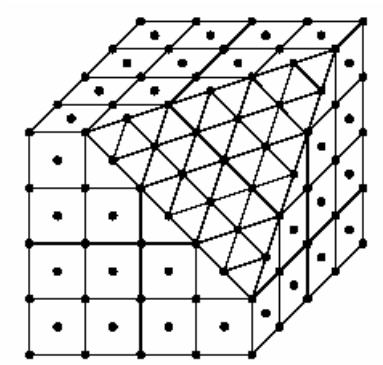


图 8 立方体面心加点后的剖面图

2.5 四面体化

本算法采用 Delaunay 三角化方法进行四面体剖分，普通意义下的 Delaunay 剖分是对一个点集的凸包进行 Delaunay 划分，使得生成的每一个四面体内部都不包含别的点。特别注意的是对多面体进行 Delaunay 剖分如果存在多于 4 个点共圆的情况，则可能出现非常薄的切片。相对于二维的 Delaunay 剖分的好的性质来说，对 3 维或更高维体进行 Delaunay 剖分时，不存在这些好的性质。在很多情况下，四面体剖分的质量可以通过破坏 Delaunay 准则来得到改善。四面体剖分的步骤包括：

2.5.1 初始波前面的选定

四面体剖分时，本算法从多面体的表面三角形开始，逐层向内部推进；由于采用了 Delaunay 准则来选择构成四面体的点，所以和一般波前法不同的是算法结果和选择面的顺序没有关系。同时对于当前三角形来查找第四个点来构成四面体时，用八叉树结构来进行加速。本算法选定位于外表面上的所有三角形为波前面。

2.5.2 四面体生成

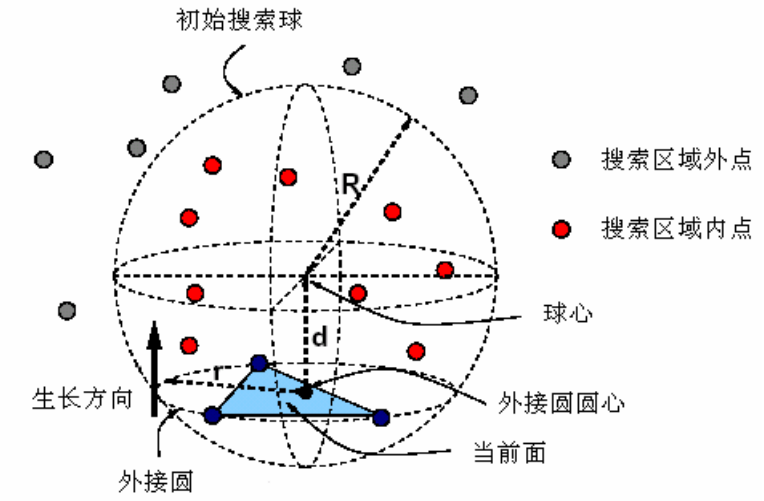


图 9 当前面初始搜索区域示意图

首先计算当前面的外接圆的圆心和半径 r，沿着当前面生长的方向从外接圆圆心前进 2r, 得到搜索球的圆心，搜索球的初始半径定为 $\sqrt{5}$ r。初始搜索区域如图 9 所示。如果搜索不到点则把搜索半径扩大，继续查找。对找到的位于搜索区域内的点算法进行如下处理：

删除位于当前面搜索方向后面的点和位于三角网格上但是不位于波前面上的点；
for 所有候选点 do
　　去掉位于当前面和选定候选点确定外接球外的点；
end for

如果最后有多个点共球，则选择形成四面体形状最好的一个候选点(Delaunay 点)。把生成的四面体加入到生成四面体列表中，更新波前面。当前面从波前面中删除，分四种情况向波前面中添加边和三角形。

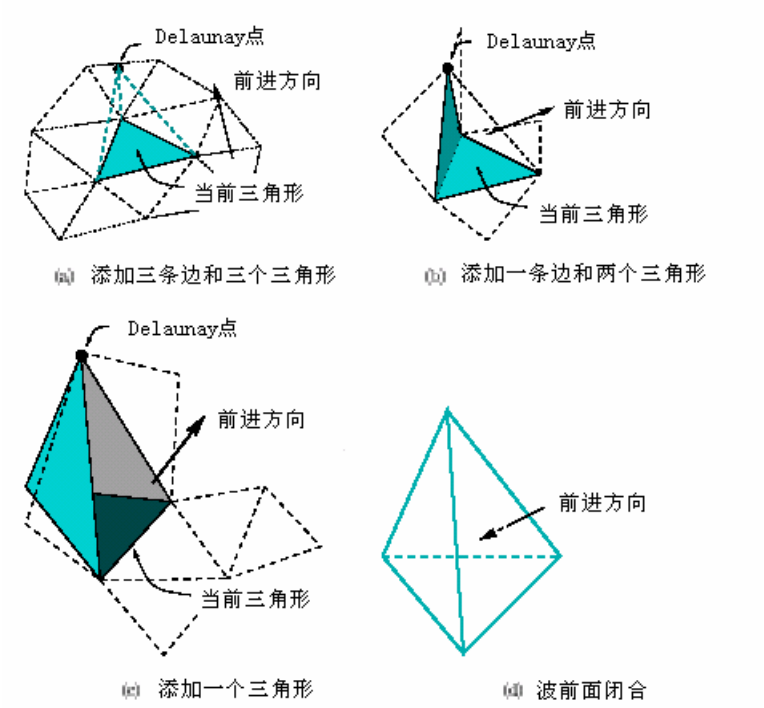


图 10 四种波前面的添加情况示意图

Delaunay 点不是当前三角形一临接三角形顶点，添加三条边和三个三角形，如图 10(a)；
Delaunay 点是当前三角形一临接三角形顶点，添加一条边和两个三角形，如图 10(b)；
Delaunay 点是当前三角形两个临接三角形顶点，添加一个三角形,如图 10(c)；
Delaunay 点是当前三角形三个临接三角形顶点，波前面闭合,如图 10(d)。

如果波前面为空，则算法结束，否则从波前面中取另一个三角形作为当前三角形。

3 系统设计 & 数据结构

主要类之间的关系（UML 描述）：

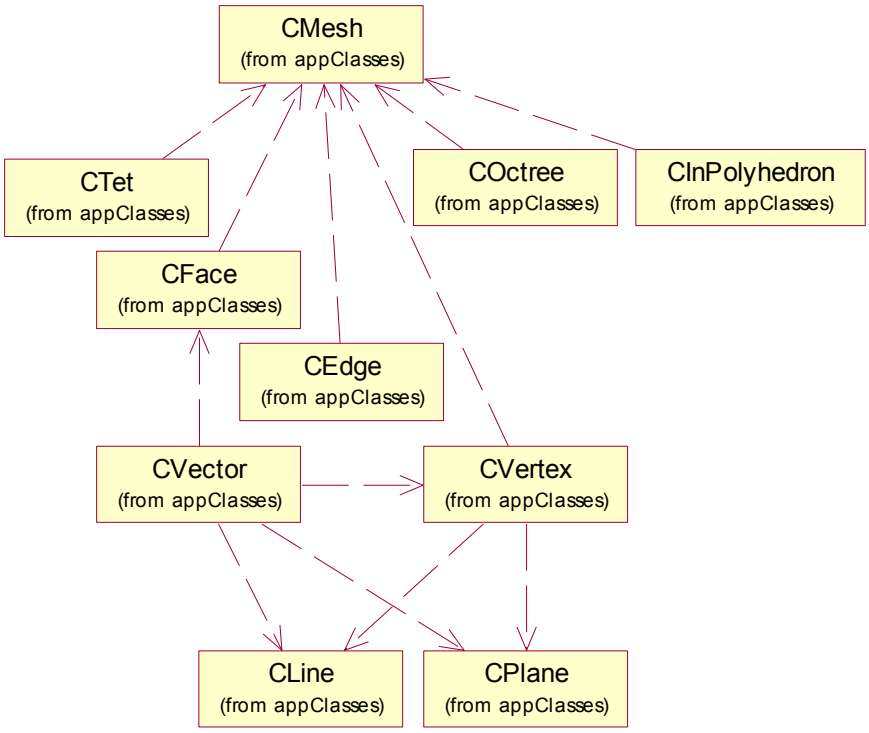


图 11 类图

CMesh 对象包含顶点数组、面数组、边数组、四面体数组；每个四面体关联包含的面，每一个面有两个以它为面四面体的索引；每个面有包含的边的索引、顶点的索引、法向；每个边有以它为边的面的索引；每个边有它顶点的索引。

```
class CMesh
{
    …
public:
    void octreeDivide();                    //八叉树的空间划分
    BOOL meshAdjust();                      //表面三角剖分的调整
    void actionBeforeTriangulate();         //插入点
    int meshTriangulation();                //四面体剖分

    vector <CVertex> m_ArrayVertex;         //顶点数组
    vector <CEdge>   m_ArrayEdge;           //有向边数组
    vector <CFace>   m_ArrayFace;           //面片数组
    vector <CTet>    m_ArrayTet;            //四面体数组
    CVertex m_lower, m_upper;
    COctree *m_Octree;
    CInpolyhedron* m_pJudgepoint;
};
```

顶点类（CVertex）是最基本的类，记录三维空间点的坐标。

```
class CVertex
```

```
{
    …
public:
    double     m_Coordinate[3];                    //顶点的空间坐标
    vector <long>     m_ArrayFaceNeighbor;    //共顶点的面的索引数组
    CVector *m_Normal;                            //顶点的法向量（用于绘制）
};
```

有向边类（CEdge）由边的起点和终点的顶点索引定义。

```
class CEdge
{
    …
public:
    long m_Start, m_End;          //边的起点和终点的顶点索引
    long m_Face1, m_Face2;      //表面上共边的面片索引（1左面、2右面）
};
```

面片类（CFace）是记录空间拓扑关系的基本单元。

```
class CFace
{
    …
public:
    long m_Vertex[3];             //面的三个顶点索引（逆时针方向）
    long m_Face[3];               //三个相邻的面索引
    long m_Edge[3];               //面的三条边索引
    CVector   m_Normal;           //面的法向量
    int m_Tet[2];                 //共面的四面体索引（0正面，1反面）
};
```

四面体类（CTet）

```
class CTet
{
    …
public:
    int m_face[4];              //构成四面体的四个面索引
};
```

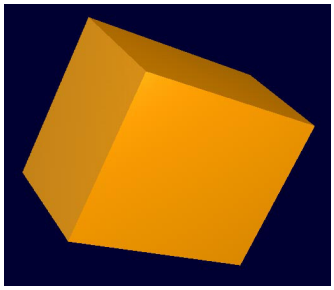
向量类（CVector）由空间的三维坐标定义向量，并重载了基本的向量运算。

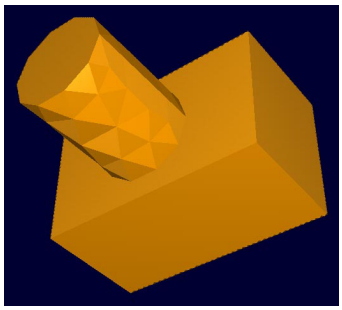
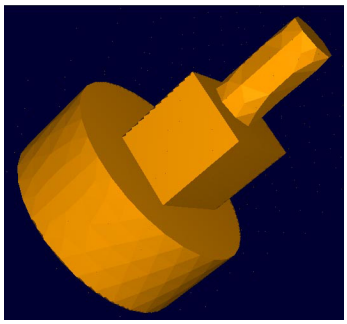
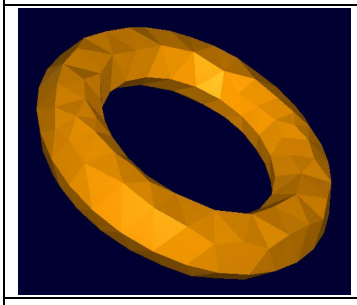
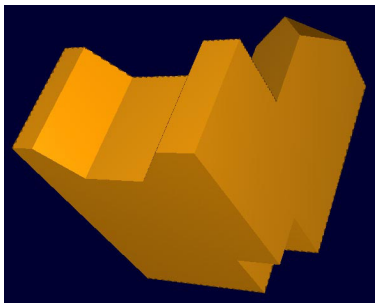
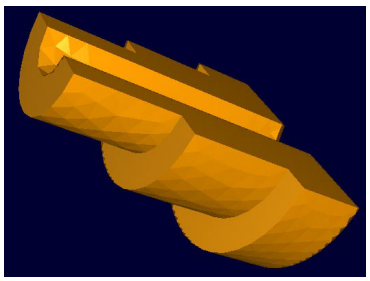
```
class CVector
{
    …
public:
    double x, y, z; //空间坐标
    CVector operator + (const CVector& v) const;     //矢量加
    CVector operator - (const CVector& v) const;     //矢量减
    double    operator * (const CVector& v) const;     //矢量乘(点积)
    CVector operator ^ (const CVector& v) const;     //矢量乘(叉积)
```

```
        CVector operator * (double num) const;          //数乘
        CVector operator / (double num) const;          //数除
        CVector operator - () const;                    //单目减
        int operator == (const CVector& v) const;       //判等(不等)
        int operator != (const CVector& v) const;
        double& operator [] (int i);          //取元素
        void SetUnit ();                      //设为单位矢量
}
class COctree
{
public:
        …
        int m_Layer;                          //本单元所在层数
        int divide(CMesh *m_Mesh);            //递归剖分函数
        COctree* m_Son;                       //儿子节点
        COctree* m_Father;                    //父节点
        char    m_Label;                      //'o'外部,'b'相交,'i'内部
        vector <int> m_ArrayVertex;           //本八叉树单元包含的顶点
}
class CInpolyhedron
{
        …
public:
        char InPolyhedron(double x,double y,double z);     //判断一个点在多面体内外
        tPointd bmax;                                      //体的包围盒右上角
        tPointd bmin;                                      //体的包围盒左下角
        tPointd Box[PMAX][2];                              //体的每一个面的包围盒
        CMesh *m_pmesh;                                    //关联的体
}
```

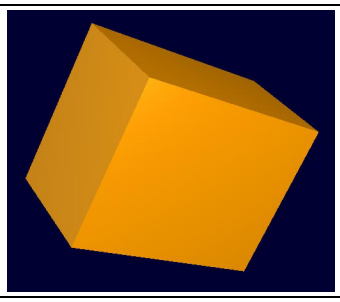
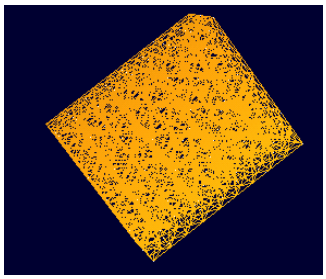
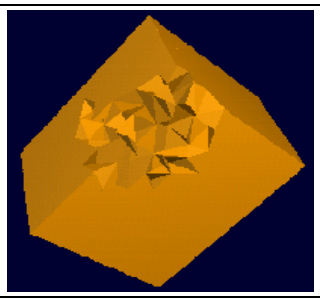
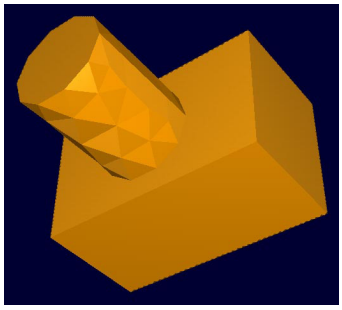
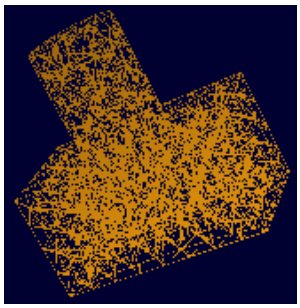
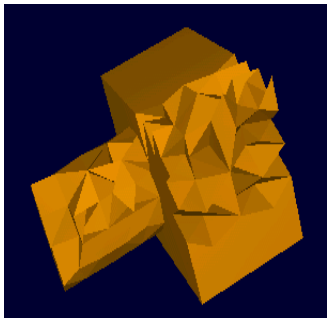
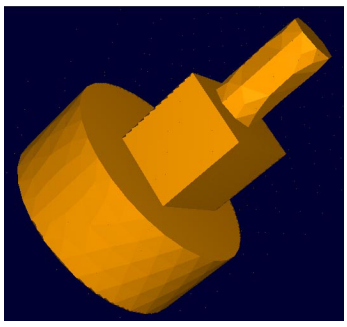
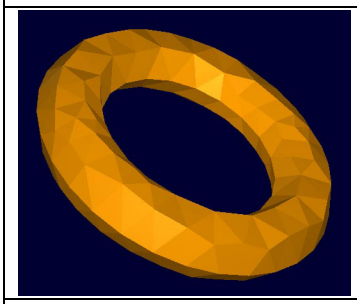
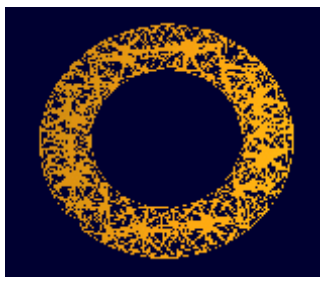
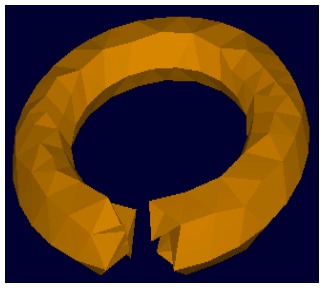
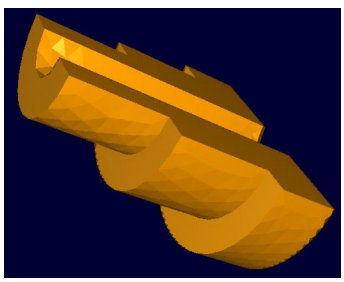
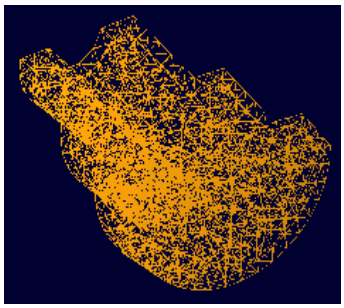
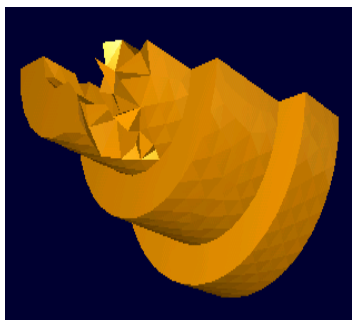
4 测试、对比结果

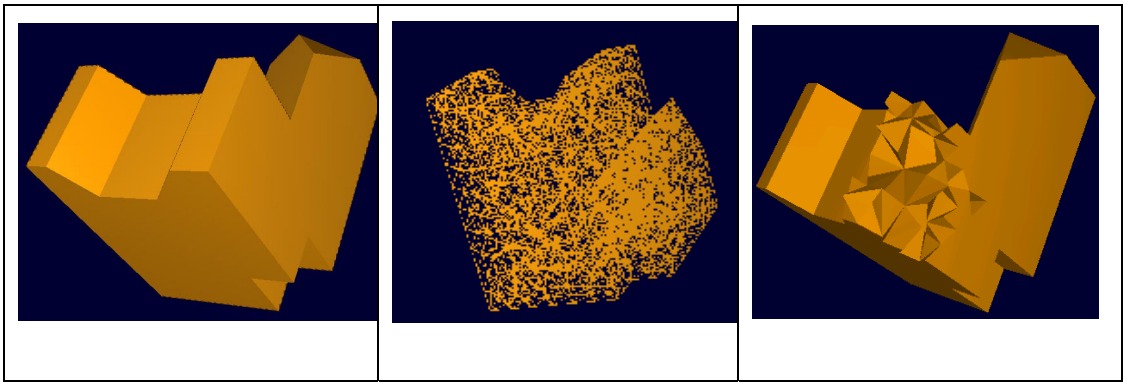
我们对六个例子列出了原始数据和结果数据中的结点数，面数以及四面体数。

实验数据	原始数据（表面三角剖分）		结果数据（四面体剖分）		
	结点数	面片数	结点数	面片数	四面体数
	338	672	573	5461	2663

	253	502	389	3365	1612
	428	852	510	3852	1779
	192	384	307	2594	1201
	591	1178	722	5580	2533
	584	1164	676	5005	2228

对这六个例子我们分别用线框图和实体图来显示生成的四面体的结构。这六个例子生成的四面体结构图如下：

实验数据	四面体结构（线框图）	四面体结构（实体图）
		
		
		
		
		



5 实现过程中遇到的问题及解决对策

5.1 对初始三角网格的调整

因为本算法的第一步，我们需要对初始的三角网格进行调整，使得三角网格满足初始的密度函数。因为三角面片重构（remesh）是一个很复杂的课题，实现 remesh 不论时间上还是人力上我们都不可能在短时间内做得很好，所以我们借助了 Ansys™来实现 remesh。然后利用 Ansys™的中间结果进行进一步的处理。

5.2 判断点在四面体内外

由于在空间的八叉树划分时给八叉树加标记和内部点添加时都要判断一些点是否在四面体内部。我们从 www.cs.smith.edu/~orourke/books/compgeom.html Computational Geometry in C 网站上下载了源码。由于下载的源码只能对整点的多面体进行点在四面体内外的判断，我们对原算法进行了大幅度的修改，组织成 Inpolyhedron 类，并作为 CMesh 类的一个成员。使得对于浮点数表示的多面体也能进行点在多面体内外的判断。

5.3 点在四个点外接球内外的判断

由于我们对当前面查找 Delaunay 点时要用到点在四面体内外的判断。由于在 Gems™里面有了 CLine 和 CPlane 类的定义和实现。所以我们借用了 Gems™里面的功能，两条边的垂直面的交线和第三条边垂直面的交点的球心，球心和任一点的距离为外接球的半径。计算几何上我们学到了判断一个点在三个点外接球的矩阵计算公式。计算公式如下：

$$inSphere(a,b,c,d,p)=\begin{vmatrix} a_x & a_y & a_z & a_x^2+a_y^2+a_z^2 & 1 \\ b_x & b_y & b_z & b_x^2+b_y^2+b_z^2 & 1 \\ c_x & c_y & c_z & c_x^2+c_y^2+c_z^2 & 1 \\ d_x & d_y & d_z & d_x^2+d_y^2+d_z^2 & 1 \\ p_x & p_y & p_z & p_x^2+p_y^2+p_z^2 & 1 \end{vmatrix} > 0$$

图 12 判断一个点在四个点外接球内外的判断

我们发现用计算几何里的公式判断，需要 5!*4 次乘法运算和 5!-1 次加法运算及一次比较运算。而如果我们计算出来了球心和半径以后的操作只需要 1 次乘法 1 次加法和一次比较运算即可。所以如果对多个点进行点在四个点外接球内外的判断时，计算圆心和半径要比用这个行列式判断优的多。

6 结论与展望

算法对一些工业模型具有良好的鲁棒性，能适应工业上有限元分析对一些规则工业模型的剖分需求。由于利用了 Delaunay 性质，剖分的四面体质量较好，避免产生的狭小切片。本算法为启发式算法，其复杂性与初始面数、体的结构以及内部四面体的密度函数都有关系。对于每一个四面体，最坏情况下其搜索范围可能包含了整个的 n 个点，需要搜索 O(n)个点。所以生成整个四面体网格时间复杂性为 $O(n^2)$，n 为生成的四面体数目。

本算法对一般模型处理时，不仅要在体的内部加入附加点，还要对原始模型进行调整。这使得算法的应用受到一定的局限性。由于任何模型在加入附加点都存在四面体剖分，所以一个不改变原始模型，但是能有效保持原来体拓扑结构的四面体剖分算法将会大大改善本算法的实用性。

7 操作说明

7.1 模型的打开和关闭

在 File 菜单中用 Open 可以选择打开一个模型文件。用 File 菜单中的 Close 可以关闭此模型。如果在已经打开一个模型的情况打开新的模型必须先 Close 当前模型然后打开新模型。

7.2 模型的显示

从 View 菜单->Object 菜单中可以选择模型的现实方式：

AS Vertex	以点的方式显示；
AS Framework	以线框的方式显示，如果没有四面体剖分完，则只显示模型的线框图，四面体剖分结束后，显示四面体剖分的线框图；
AS Surface	以面的方式显示；
Switch	显示开关。

从 View 菜单->Space Division 菜单可以选择八叉树的现实方式：

AS Nodes	以点的方式显示八叉树；
AS Edges	以线框的方式显示八叉树；
Switch	显示开关。

从 View 菜单->Tetrahedra 菜单可以选择以四面体方式显示。

在程序的状态条上显示了当前的操作，进度条显示执行过程；剖分结束后在状态条上显示了所有四面体的顶点数，边数，面数以及四面体数。

7.3 模型的操作

按住鼠标左健移动，可以旋转模型；按住鼠标右健移动，可以放缩模型。

在四面体显示方式下（View 菜单->Tetrahedra），按住 CTRL+鼠标左健可以选择删除四面体，查看生成的四面体结构。

最后感谢邓俊辉老师和实验室的师兄们在算法思想和实现方面的帮助。邓俊辉老师在算法可行性方面给了我们很大的指导。张晓鑫师兄在我们 Ansys™安装使用方面提供很大的帮助。刘玉身师兄在点在多面体内外判断方面给了我们很大的帮助，在此感谢他们。

参考文献

[1] R. Radovitzky, M. Ortiz, Tetrahedral Mesh Generation Based on Node Insertion in Crystal Lattice Arrangements and Advancing-Front-Delaunay Triangulation, Comput. Methods Appl. Mech. Engrg. 187(2000) 543-569.

[2] S. J. Owen, M. L. Staten, Q-Morph: An Indirect Approach To Advancing Front Quad Meshing, Int. J. Numer. Meth. Engng. 44, 1317–1340 (1999).

[3] Anshuman Razdana, MyungSoo Baeb, A hybrid approach to feature segmentation of triangle meshes, Computer-Aided Design 35 (2003) 783–789.

[4]S.Owen, A Survey of Unstructured Mesh Generation Technology
http://www.andrew.cmu.edu/user/sowen/survey/.

[5]计算几何讲义，邓俊辉