

深入理解数值计算网格(全篇)

原创 邓子平 [多物理场仿真技术](#)



为了方便阅读，将所有文章放在了一起，并且点击标题可以单独阅读。

1. 网格介绍

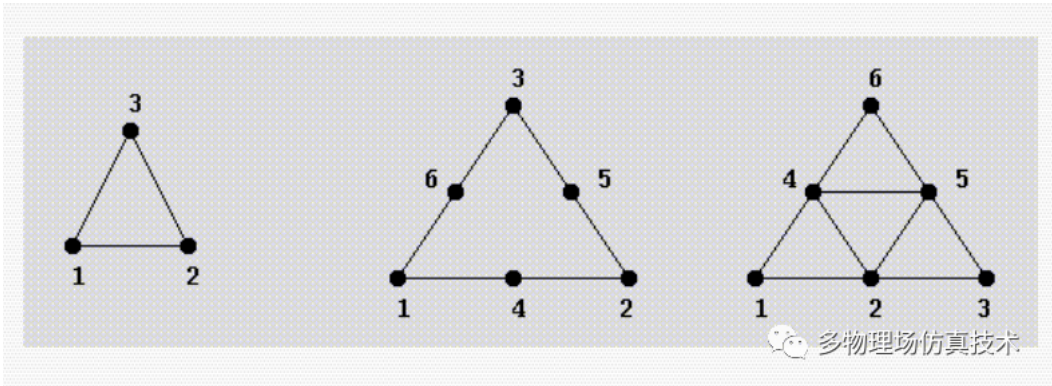
网格(Mesh)在数值计算中有着举足轻重的作用。主流的数值仿真方法诸如有限元，有限体积，有限元，边界元都是以网格为计算对象。而差分法等，时域有限差分等也是以网格(Grid)点为计算对象。

什么是好网格？

网格的好坏直接决定了仿真计算能否成功，以及正确性，精度，性能。
简单的说就是尽可能用最少的网格，最真实的反应物理量的变化规律。

常见的网格种类：

1. 三角形(Triangle)



以上图形从左往右分别为

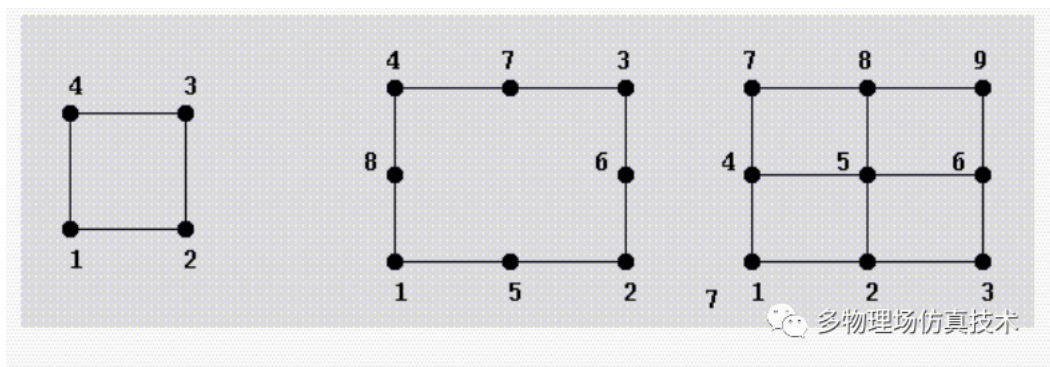
左> 0阶一次单元，3个顶点，3条边

中> 1阶二次单元，6个顶点，3条边

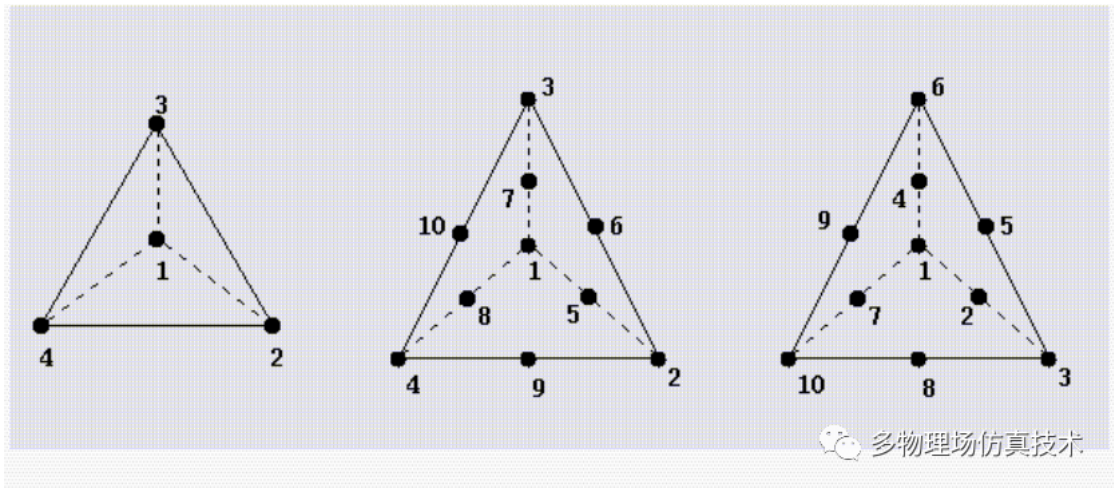
右> 拉格朗日1阶二次单元，6个顶点，6条边

以下单元类型可以类推

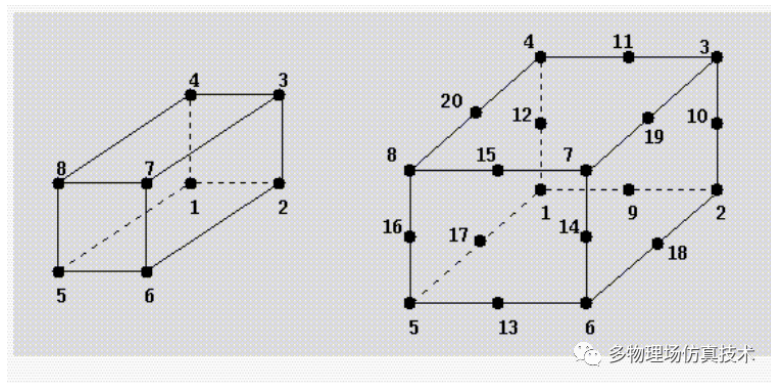
2. 四边形(Quadrilateral)



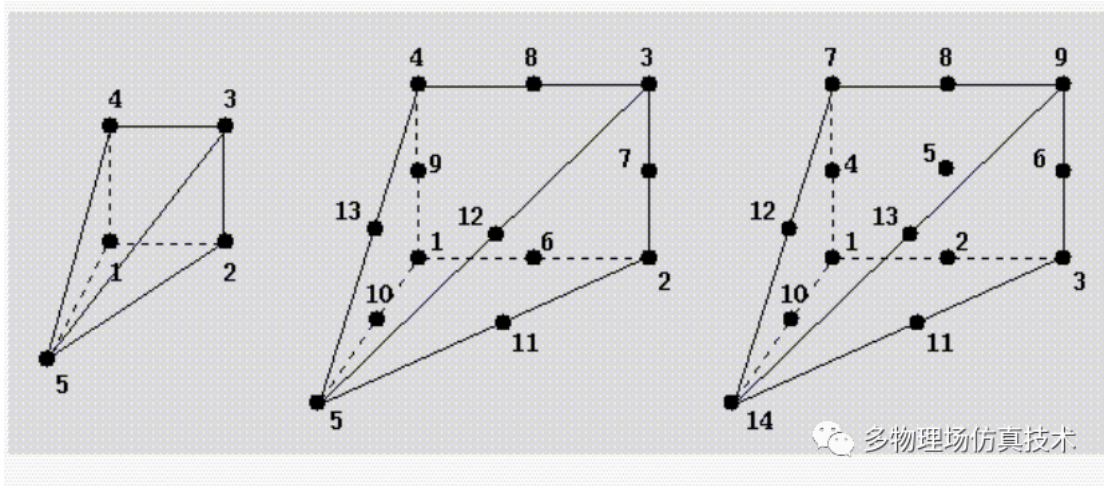
3. 四面体(Tetrahedron)



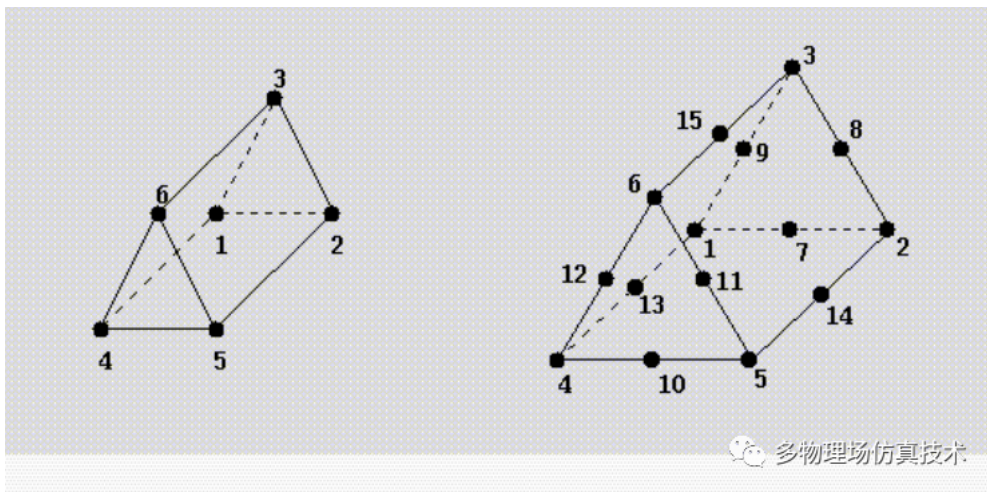
4. 六面体(Hexahedron)



5. 金字塔单元(Pyramid)



6. 楔形单元(Wdge)



网格的分类：

网格对应的英文有两个Mesh和Grid，Grid 的基本意思为格子，即四四方方非常规整的网格。

对于Mesh，我们通常分为**结构化网格**和**非结构化网格**。理解很简单，除了**四边形**和**六面体**是结构化网格，其它都是非结构化网格。

对于Grid，通常情况下我们用作有限差分方法的网格。

高阶单元：

除了上述我们提到的**0阶1次单元**以及**1阶2次单元**，在数值计算上还有更高阶的单元。高阶单元在数值计算上具有更高的精度，当然也会带来更大的计算量。在所有物理场的数值计算中，通常情况下**1阶2次单元**具有较好的性价比，即在精度和计算时间上有比较好的平衡。

以电磁矢量棱边3维四面体单元为例，其基函数自由度DOF与阶次的关系为：

$$\text{DOF} = (n+1)(n+3)(n+4)/2; \quad n \text{ 为阶的数值}$$

$$n=0 \text{ 时, } \text{DOF}=6$$

$$n=1 \text{ 时, } \text{DOF}=20$$

$$n=2 \text{ 时, } \text{DOF}=45$$

$$n=5 \text{ 时, } \text{DOF}=216$$

以电磁矢量棱边3维六面体单元为例，其基函数自由度DOF与阶次的关系为：

DOF = 3(n+1)(n+2)(n+2)

n=0时, DOF=12

n=1时, DOF=54

n=2时, DOF=144

n=5时, DOF=882

也就是说对于5阶的六面体单元，其一个单元的刚度矩阵可达882*882，对于开发验证，复杂度远远高于常用的0阶和1阶单元。

几何和网格

在网格之前，我们通常拿到的都是几何数据，比如BREP结构，STL面片结构，或者参数化的几何结构。

几何离散成的三角形称之为“面片”，非网格，大部分显示引擎底层都使用三角形来渲染对象，因此2维、3维几何都需要三角化（称之为“面片化”），比如一个长方体共有6个面，每个面需要离散成两个三角形，总共12个三角形。这12个三角形我们称之为12个“面片”，而非网格！一般情况下“面片”质量很差。“面片”有两个功能：一是用来做显示渲染数据；二是可以作为网格划分的输入数据，用来生成面网格。

对于隐式曲线曲面，需要设置合适的离散参数，从网格端进行解析。

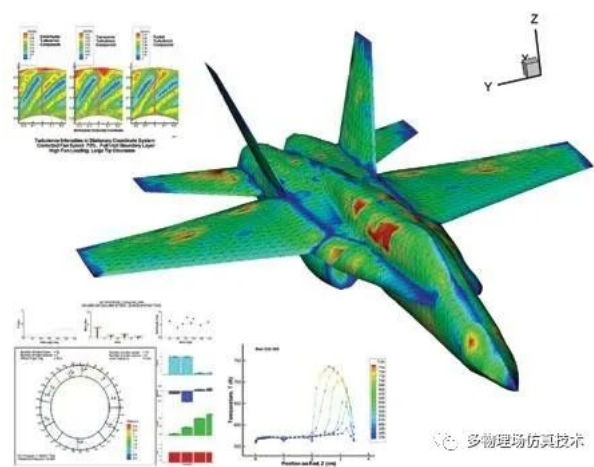
有限元方法中的单元和网格

网格是几何上的概念，单元则是数值方法中的概念。

弹簧只能在一个方向上发生变形，是典型的1维单元；同理壳单元（shell）需要XY两个方向来定义，是2维单元；四面体，六面体是3维单元，也称为实体单元；对象可以看做质点的为0维单元，比如称之为“定楼梯球”的调谐质量阻尼器。

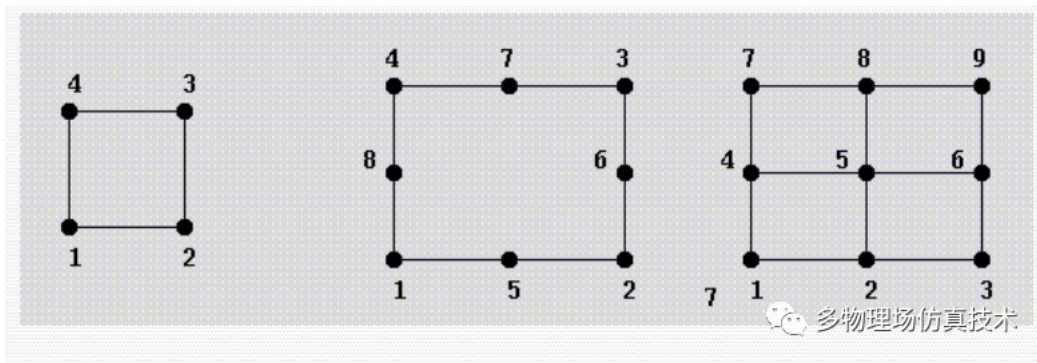
网格生成算法

网格生成算法无论是在理论还是实际应用中都比较成熟，后续将会详细介绍。在这个领域很难有颠覆式的发展，未来网格底层生成算法可以和实际工业应用更加紧密结合，优化网格生成效率。

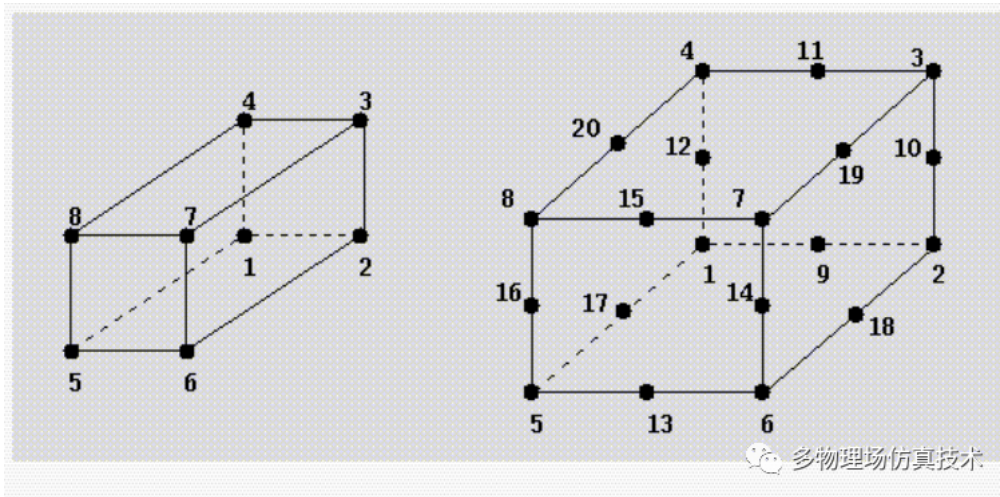


2.结构化网格生成算法

结构化网格主要指四边形和六面体



常用四边形网格



常用六面体网格

在实际应用中，一方面结构化网格在形函数基函数计算，等参变换，处理模型，模型编辑上有优势，比如网格分割，组合，沿固定方向加密，还可以方便拆成非结构化网格；另一方面，对于不太规则几何的几何体，自动生成结构化网格比较困难，需要大量的人工干预。所以在通用有限元分析中一般优先推荐结构化网格，另外CFD分析中，六面体网格天然适用流体的有限体积计算方法，也是首选。

1. 最简单的结构化网格方法是扫掠。对于三维，选中一个面，划分四边形，然后沿另一维度扫掠生成六面体。这种方法需要用户手工参与，且对几何有一定要求，扫掠生成的网格质量一般都不错。除了单向扫掠，实际算法中会有多轴扫掠。扫掠的改进方法即类似于在扫掠方法渐进的波前法，又叫plastering方法。

2. 结构化网格生成算法中，有一种间接生成方法，即首先创建非结构化网格，再合并。比如二维网格中先产生三角形，然后三角形两两组成一个四面体。六面体则是先生成四面体，然后组装六面体。这种方法又叫Morph算法。

3. 由于结构化网格有比较好的边界特性，所以多块(MultiBlock)划分也是常用的方法。即首先将几何区域分成子区域，在子区域里分别划分网格，再进行合并。比如生成六面体，一般会将空间进行八叉树划分，在每个八叉树子节点内进行划分，但这种方法需要处理边界，优化边界网格的质量。

4. Whisker Weaving是一种基于拓扑连接的自动六面体生成算法。首先在边界上生成四边形，然后以四边形为输入采用波前法推进在三维模型内部建立六面体的对偶表达，根据对偶表达生成六面体。

全自动的六面体网格生成仍然是网格生成的难点。

考虑到结构化网格生成的工程复杂性，本文仅对生成算法做基本介绍。通用前处理软件以及各种仿真软件中都会提供结构化网格功能，但全自动的网格质量一般都不会太好，需要一定的手工操作。

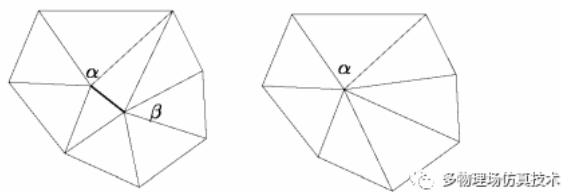
网格优化

网格生成之后，最后需要优化网格质量，常见的几种质量优化方法：

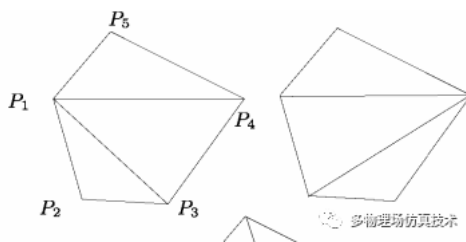
1. 移动顶点位置

移动顶点位置，让和顶点关联的单元质量都能改进

2. 移除顶点(通常说的edge collapse)



3. 交换边



4. 交换面(同交换边)

5. 分解边

即将一条长边分解成两条短边

最后说一下网格的并行计算，网格并行主要有两点：一是几何模型的分解或者网格模型的分解；二是网格的拼接。这两点做好后，网格大规模并行计算基本没什么问题。现在的分布式计算和并行计算工具已经非常成熟，可以拿来直接使用。

3.非结构化网格生成算法

通常说的非结构网格主要指非四边形和六面体网格，包含三角形，四面体，楔形，金字塔等，在实际应用中最常用的还是三角形和四面体。本文也主要介绍三角形和四面体的生成算法。

非结构化网格自动生成主要包含三种方法：

1.Delaunay method

(关于此方法的汉语翻译特别杂，不建议用中文翻译)

2.Advancing-Front method (波前法)

3.Spatial decomposition based method

(基于空间分解方法)

1.Delaunay method

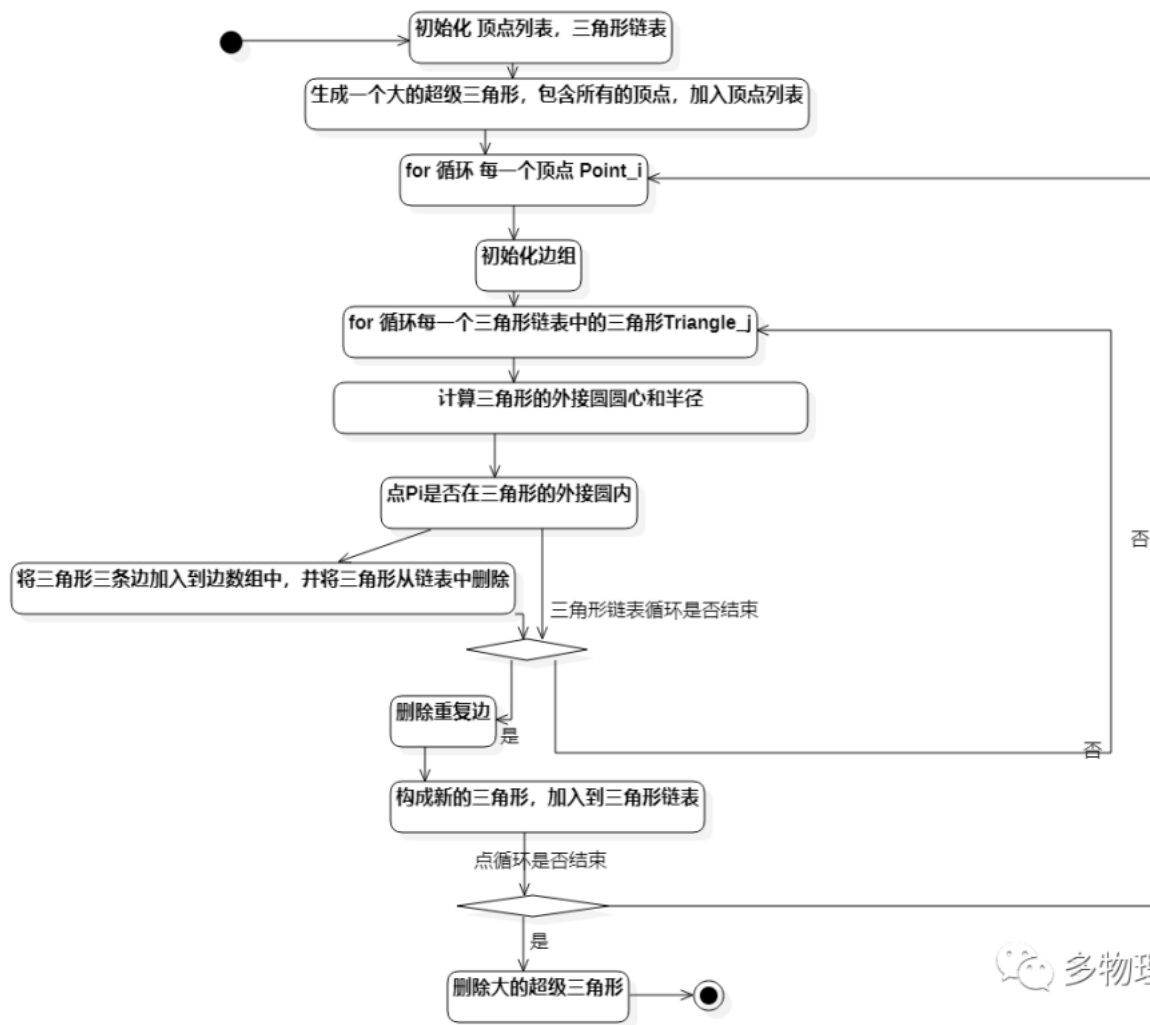
生成三角形最常用的方法是Delaunay方法，也是所有网格生成算法中的最“基础方法”，在很多开源软件中都有其方法的实现。

Delaunay三角网格定义：平面上的点集P是一种三角剖分，使得P中没有点严格处于剖分后中任意一个三角形外接圆的内部。简单讲，给出一堆离散点，Delaunay网格是满足所有网格质量最好的网格。

Delaunay方法目前有很多改进方法，只介绍其中最基础的一种方法:逐点插入法

生成算法：

- 1.给出一堆离散点，初始化三角形链表
- 2.给出一大三角形包含所有离散点
- 3.计算三角形链表中的每一个三角形的外接圆圆心和半径
- 4.如果点在已知三角形外接圆内，则把三角形的三条边加入到边数组中，在三角形链表中删除该三角形
- 5.删除所有重复的边，重复步骤3
- 6.用点和边数组中的每条边都组合成一个三角形，加入到三角形链表中，重复步骤2



Delaunay网格插入点UML活动图

作为基础的网格生成算法，Delaunay不仅支持二维，而且也支持三维。在三维计算中，只需将外接圆的判断改为外接球的判断即可。

2.Advancing-Front method（波前法）

Advancing-Front 方法又叫波前法，前沿推进法等。

其核心思想是沿着原始的网格边界生成网格，然后逐步推进生成，类似于波浪向未划分网格区域前进，直到所有区域被网格填满。

波前法思路清晰，尤其适用于已知边界的情况，其网格属性在生成过程中可以动态调整，控制好，最终生成的网格质量也比较好，是生成非结构化网格的基础算法之一，同时Advancing-Front 方法的经过改进，也适用于四边形和六面体等结构化网格生成。

经典的波前法对二维从一组边开始，对三维从一组三角形面开始。以二维为例，网格生成策略一方面逐个单元逐个单元生成，创建插入新点和新单元，同时新的单元和之前的单元保持连通性。算法的一个关键点是如何插入新的点，新的单元需要满足网格尺寸和单元质量要求，波前法的一个优势是启发式（heuristic）算法，可以生成高质量，梯度可控的网格。相比其它算法，边界的完整性可以得到保证。其不足是在三维方法时，有可能出现不收敛的情况，即有些区域很难完全填充网格。

波前法的典型流程：

1. 定义网格输入数据和参数，包括整几何区域，边界，网格尺寸，网格梯度
2. 离散边界
3. 从边界任意位置开始，逐步插入点，形成单元，迭代指导求解区域被网格填满

在这个过程中，如何插入点和形成单元是算法最核心的部分，其涉及到：

1. 前进方向的单元选择
2. 前进分析以及定义最优点需要考虑的各种可能性
3. 单元构建有效性分析
4. 选择合适的数据结构

相比较其它方法，波前法在三维实现上要远远高于二维。三维需要考虑更多的因素和特殊情况，同时在算法效率上，三维的要求也更高。

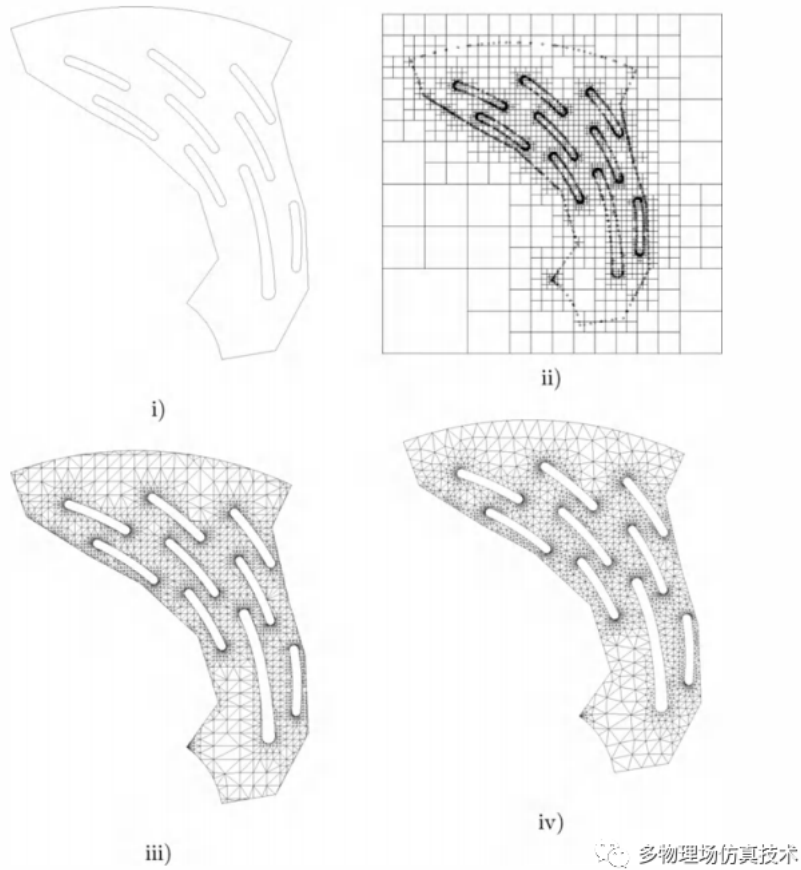
3.Spatial decomposition based method

(基于空间分解方法)

基于空间分解方法相对简单，主要用到空间树结构，即二维四叉树和三维八叉树。以平面四叉树为例：

1. 根据几何对象，创建合适的树边界
2. 根据对象点的分布，创建四叉树结构
3. 在每个四叉树内部，利用四叉树的边和点，生成网格
4. 调整网格，使其满足网格质量要求和梯度分布

一个典型的例子如下：



对于使用四叉树和八叉树方法生成网格，两个核心点：

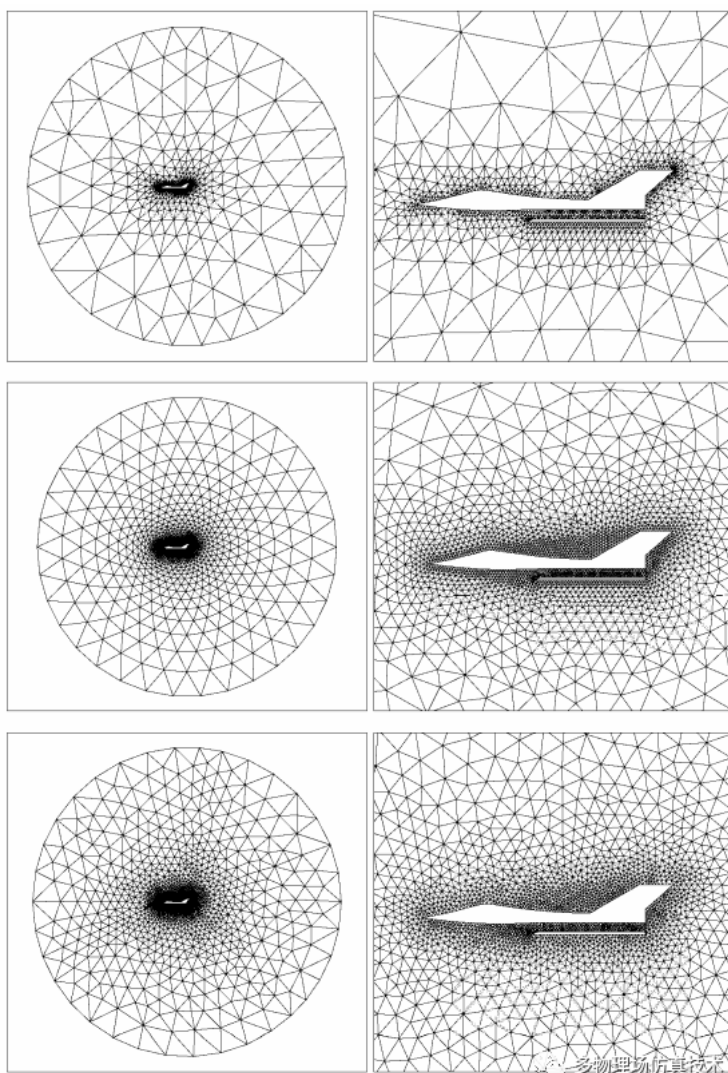
1. 动态生成不平衡的树，即每个同级的树结点数目并不相同；
2. 在树结点内部选择合适的算法生成初始网格

笔者在几何，图形，网格，求解器开发中，都较多的使用到了八叉树结构，八叉树结构可以说是一种基础性的数据结构和方法。

三种方法生成的网格比较：

method	np	ne	Q_M	Q_{worst}
quadtree	1,246	2,171	1.25	1.88
advancing-front	2,557	4,795	1.1	1.61
Delaunay	2,782	5,528	1.16	1.82

np 为点的数目， ne 为单元的数目， Q_m 为网格的质量， Q_{worst} 为最差的网格质量



图片从上往下方法分别是：

二叉树,波前法,Delaunay

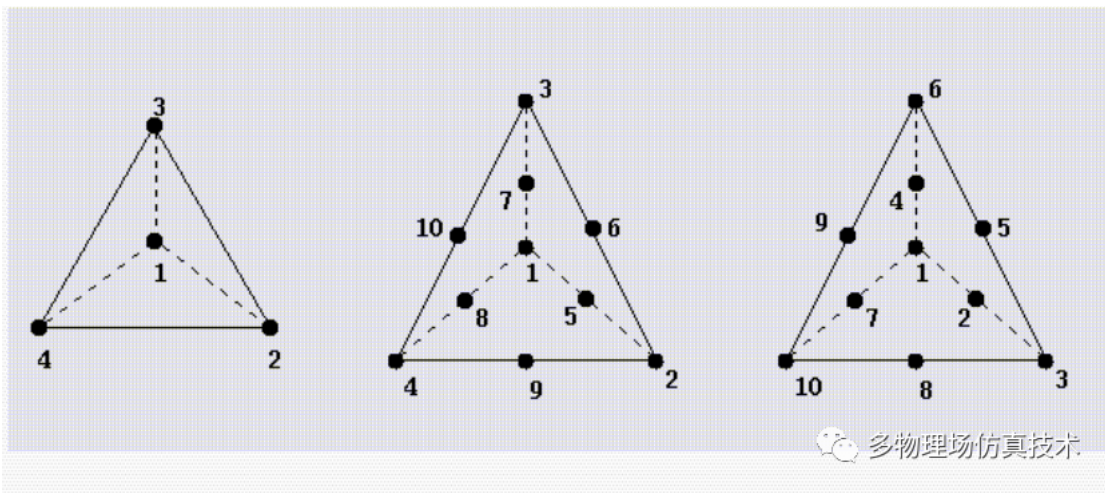
图片皆来自于书 Mesh Generation (Application to Finite Elements)，考虑到网格参数设置的差异，重点关注网格质量和分布

在实际应用中，三种方法并不是独立存在，经常会有方法混用的情况，比如在CFD网格中，不同区域采用不同的网格策略，可能效率和稳定性更好。

4.万能的四面体

在三维分析中经常用到四面体，六面体，金字塔等三维实体网格，其中四面体因为其容易自动化生成，适应性好，数值计算方便等优点，在多物理场三维有限元分析中得到广泛应用。

几何特点




通常我们拿到四面体的是四个点的坐标，利用点可以计算出边，面的数据，在计算中要保持数据的一致性，即点，边，面的顺序要有固定规律。
以面为例，通常按照右手规则，三点顺序连接成面的法向量要一致朝外。

四面体计算中经常要用到任意四面体的体积和四面体角度


以上指标的计算方法C++源码可在 www.cae-sim.com 下方资源下载

[刷新](#)
[主页](#)
[收藏](#)
[cae-sim.com/cn/](#)


[手机书签](#)
[IBM Rational Cle](#)
[2D - OneDrive](#)
[Types of Checks](#)
[Sigrity Weekly Su](#)
[Login - Cadence](#)
[File Manager - C](#)
[Sigrity - Home](#)



FasTCAD
TCAD工具



FasMaterial
计算材料软件包



FasDICOI
CT DICOM文件浏览
[直接下载地址](#)

[源码资源下载](#)

四面体体积，角度计算

多物理场仿真技术

网格质量评估

除了常见的最大边，最小边，最大角，最小角等。还有以下四种评估指标：

1.Aspect ratio：取得最大的边长A，取得最短的高度H（高度定义为一个顶点到对面的距离）。最大边长除以高度：A/H即为Aspect Ratio

2.坍塌比：

计算公式：(H/Sqrt(A))/2.14

H为顶点到对面的高度，

A为顶点对面的三角形面积，

坍塌比范围为0到1,1是正四面体，0为完全坍塌

3.二面角

六条边，每条边关联两个面的夹角，总共有六个角

通常取最大最小的二面角评估网格质量

4.扭曲度：

单元体积与相同大小外接球条件下的理想四面体体积之比

计算公式：实际体积/理想体积

正四面体值为1

常见的四面体几何错误包括：

- 1.自由点，边，面，
- 2.重复的点，边，面和实体
- 3.四点共面
- 4.相邻两点距离或Aspect ratio超过数值计算误差

对于高阶单元，四面体网格的几何和质量计算方法相同。

网格质量是如何影响仿真精度的

几乎所有书籍解释就是网格质量差，会造成误差大，计算精度不够。

但其实网格真正对仿真精度的影响来自两方面：网络密度和网络分布。

看一个场景：假设现在网格已经最优，达到最理想状态，我们将其中两个相邻网格加密，使其网格质量变差，但其实仿真精度并不受影响，只是网格分布变差，网格密度增加。

换句话说，除去数值计算误差，网格质量并不直接影响仿真精度！而是差质量的网格造成网格整体密度不够和分布不合理！

自由度

在网格技术中，我们经常提到自由度，其实自由度是有限元的概念，而非网格，自由度是在单元上的一种属性。一个自由度表示一个计算变量。比如热分析单元每个顶点上只有一个自由度，即温度；而位移则至少有XYZ三个自由度，考虑到旋转，则有更多自由度；电磁场至少有电场XYZ，磁场XYZ六个自由度。

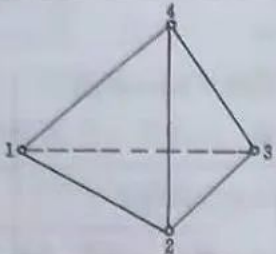
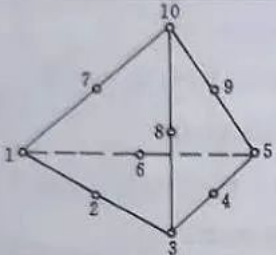
此外对于相同拓扑的网格类型，也可以根据自由度的个数来定义不同的单元，比如四面体单元在ANSYS中至少有Solid185，Solid187，Solid92等。

形函数

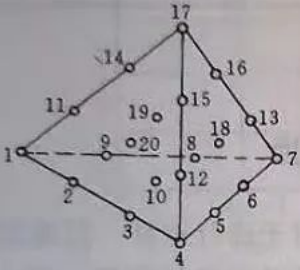
求解应力应变，以节点位移为形函数参数的单元：

一次线性单元和二次曲线单元

其中L为体积坐标，三维单元的体积坐标可以参考任意一本有限元理论方面的书籍。

4结点 线性四面体元		u, v, w	$N_1 = L_1$ $N_2 = L_2$ $N_3 = L_3$ $N_4 = L_4$ $(L_i \text{ 为体积坐标})$	计算简单, 可适应复杂 几何形状, 单元内应力 为常量,精 度较低
10结点 二次四面体元		u, v, w	$N_1 = L_1(2L_1 - 1), \quad N_2 = 4L_1L_2$ $N_3 = L_2(2L_2 - 1), \quad N_4 = 4L_2L_3$ $N_5 = L_3(2L_3 - 1), \quad N_6 = 4L_3L_1$ $N_7 = 4L_1L_4, \quad N_8 = 4L_2L_4$ $L_9 = 4L_3L_4, \quad N_{10} = L_4$	边界可为 曲面,可适 应有尖角的 复杂几何

三次高阶四面体单元

单元名称	单元 (母单元) 形态	自由度	形函数	单元特性
20结点 三次四面体元		u, v, w	$N_1 = \frac{1}{2}(3L_1 - 1)(3L_1 - 2)L_1$ $N_2 = \frac{9}{2}L_1L_2(3L_1 - 1)$ $N_3 = \frac{9}{2}L_1L_2(3L_2 - 1)$ $N_4 = \frac{1}{2}(3L_2 - 1)(3L_2 - 2)L_2$ $N_5 = \frac{9}{2}L_2L_3(3L_2 - 1)$ $N_6 = \frac{9}{2}L_2L_3(3L_3 - 1)$ $N_7 = \frac{1}{2}(3L_3 - 1)(3L_3 - 2)L_3$ $N_8 = \frac{9}{2}L_3L_1(3L_3 - 1)$ $N_9 = \frac{9}{2}L_3L_1(3L_1 - 1)$ $N_{10} = 27L_1L_2L_3$ $N_{11} = \frac{9}{2}L_1L_4(3L_1 - 1)$ $N_{12} = \frac{9}{2}L_2L_4(3L_2 - 1)$ $N_{13} = \frac{9}{2}L_3L_4(3L_3 - 1)$ $N_{14} = \frac{9}{2}L_1L_4(3L_4 - 1)$ $N_{15} = \frac{9}{2}L_2L_4(3L_4 - 1)$ $N_{16} = \frac{9}{2}L_3L_4(3L_4 - 1)$ $N_{17} = \frac{1}{2}(3L_4 - 1)(3L_4 - 2)L_4$ $N_{18} = 27L_2L_3L_4$ $N_{19} = 27L_1L_3L_4, \quad N_{20} = 27L_1L_2L_4$	边界可为 曲面,可适 应带尖角的 复杂几何 形状

以上图片来自于朱伯芳《有限单元法原理与应用》

电磁矢量四面体单元

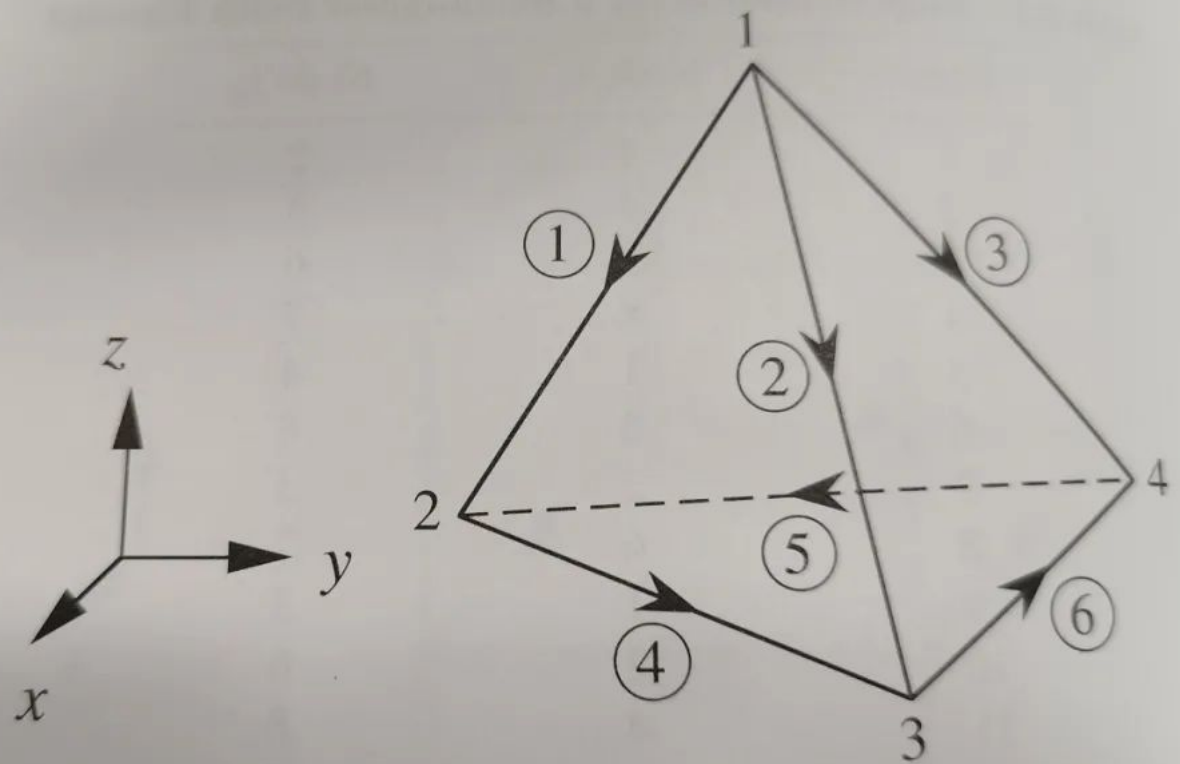


Fig. 8.12 Tetrahedral element.

多物理场仿真技术

电磁矢量四面体单元高阶形函数

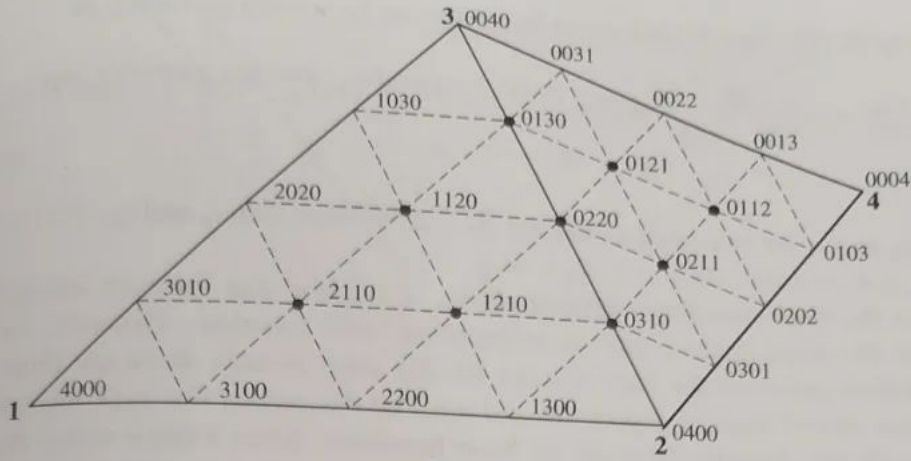


Fig. 8.28 Interpolation points on a second-order tetrahedral element for vector basis functions associated with \mathbf{W}_{23} (interior points are omitted for clarity).

B. Tetrahedral Elements. For a tetrahedral element, the zeroth-order vector basis function associated with the edge connected to nodes i_1 and i_2 is given by

$$\mathbf{W}_{i_1 i_2} = \xi_{i_1} \nabla \xi_{i_2} - \xi_{i_2} \nabla \xi_{i_1} \quad (8.113)$$

where i_1 and i_2 are defined in Table 8.2. To construct higher-order vector basis functions, we arrange the interpolation points in a pyramid format and label each point using four integers (i, j, k, ℓ) , where $i, j, k, \ell = 0, 1, \dots, p+2$, and p denotes the order of the element. This arrangement is illustrated in Fig. 8.28. The p th-order vector basis functions are then given by

$$\begin{aligned} \mathbf{N}_{ijkl}^{12} &= \alpha_{ijkl}^{12} \hat{P}_i^{p+2}(\xi_1) \hat{P}_j^{p+2}(\xi_2) P_k^{p+2}(\xi_3) P_\ell^{p+2}(\xi_4) \mathbf{W}_{12} \\ &\quad k, \ell = 0, 1, \dots, p; i, j = 1, 2, \dots, p+1 \\ \mathbf{N}_{ijkl}^{13} &= \alpha_{ijkl}^{13} \hat{P}_i^{p+2}(\xi_1) P_j^{p+2}(\xi_2) \hat{P}_k^{p+2}(\xi_3) P_\ell^{p+2}(\xi_4) \mathbf{W}_{13} \\ &\quad j, \ell = 0, 1, \dots, p; i, k = 1, 2, \dots, p+1 \\ \mathbf{N}_{ijkl}^{14} &= \alpha_{ijkl}^{14} \hat{P}_i^{p+2}(\xi_1) P_j^{p+2}(\xi_2) P_k^{p+2}(\xi_3) \hat{P}_\ell^{p+2}(\xi_4) \mathbf{W}_{14} \\ &\quad j, k = 0, 1, \dots, p; i, \ell = 1, 2, \dots, p+1 \\ \mathbf{N}_{ijkl}^{23} &= \alpha_{ijkl}^{23} P_i^{p+2}(\xi_1) \hat{P}_j^{p+2}(\xi_2) \hat{P}_k^{p+2}(\xi_3) P_\ell^{p+2}(\xi_4) \mathbf{W}_{23} \\ &\quad i, \ell = 0, 1, \dots, p; j, k = 1, 2, \dots, p+1 \\ \mathbf{N}_{ijkl}^{42} &= \alpha_{ijkl}^{42} P_i^{p+2}(\xi_1) \hat{P}_j^{p+2}(\xi_2) P_k^{p+2}(\xi_3) \hat{P}_\ell^{p+2}(\xi_4) \mathbf{W}_{42} \\ &\quad i, k = 0, 1, \dots, p; j, \ell = 1, 2, \dots, p+1 \\ \mathbf{N}_{ijkl}^{34} &= \alpha_{ijkl}^{34} P_i^{p+2}(\xi_1) P_j^{p+2}(\xi_2) \hat{P}_k^{p+2}(\xi_3) \hat{P}_\ell^{p+2}(\xi_4) \mathbf{W}_{34} \\ &\quad i, j = 0, 1, \dots, p; k, \ell = 1, 2, \dots, p+1. \end{aligned} \quad (8.114)$$

多物理场仿真技术

之前看过很多关于六面体和四面体的争论，大意上是说六面体要优于四面体，在网格选择上尽可能使用六面体。

从有限元理论上看，六面体确实只需比四面体更少的网格，就能达到更高的精度。但是对于实际工程中的模型，一方面自动化六面体生成和加密都比较困难，需要更多的手工操作，耗时耗力；另一方面六面体在很多计算中有更多限制和要求，比如自锁沙漏等现象，要人为加密网格或者缩减积分，需要一定的工程经验；最后各个领域的要求不太一样，一般CFD流体计算中六面体偏多，在无法生成六面体的地方需要混合四面体网格，热仿真对网格拓扑要求不高，电磁很少用到六面体，一般的争论在结构仿真中，而结构仿真大多一般用1,2维单元，体单元在局部位置分析较多。

综上所述，其实并不存在六面体和四面体哪个更好的问题，只是对于实际的工程问题，基于工程师的经验，计算资源，仿真类型，需要选择合适的单元。

主流开源，商业网格引擎都提供了对四面体的支持。包括Gmsh, Netgen, tetgen, phg, CGAL, VKI, meshgem, Simmetrix等等。

另外有过一份关于全球公开的四面体网格开源和商业软件的总结，并不是最新。

复制链接到浏览器中查看：

<http://www.robertschneiders.de/meshgeneration/software.html>

5.网格参数

网格参数是网格的一个重要因素，包括网格生成策略，网格生成参数，网格属性，以及网格质量评价参数。除了之前介绍的网格生成算法，网格参数决定了网格数据的最终质量，密度，梯度等指标，也直接决定了网格能否划分成功。这也是数值计算网格和几何三角化最大差别所在。

1.网格生成参数

网格生成参数涉及到网格生成时采用的参数，

每一个参数都有以下方面的内容：

1. 参数内容--在网格划分中需要取得的具体数据
2. 参数属性--参数的属性提供了网格的各种属性，这些属性给网格划分中各种策略处理提供依据
3. 优先级--优先级则是在和其它参数相互冲突时，如何处理

以我们最常见的网格尺寸为例。

1. 参数内容：用来指定使用的网格尺寸大小；
2. 参数属性：指定是整体尺寸，还是局部尺寸；是绝对尺寸还是相对尺寸；尺寸的单位；是否是指定几何对象或特定属性的尺寸等等
3. 优先级：一般来说，局部尺寸优先级高于整体尺寸，网格尺寸优先级高于指定的网格数量。

所有的软件网格划分参数都是以这些为基础，在上层应用中给出部分接口和参数设置，或者根据经验进行优化组合后暴露给用户少量接口，比如各种软件中的smart mesh size。

最常见的网格参数为

1. 整体网格尺寸

整体网格尺寸指定方式通常有：

- 1.1.直接指定全部整体网格尺寸；
- 2.1.指定模型对角线百分比，单一几何模型，如果不给出任何参数，通常取模型包围盒对角线一定比例为整体网格尺寸参数；
- 3.1.系统会根据模型几何数量，大小，给定一个默认参数，在此基础上，用户可以进行修改；
- 4.1.分级设置，按照尺寸比例按照区间分级设置，比如1-5级。

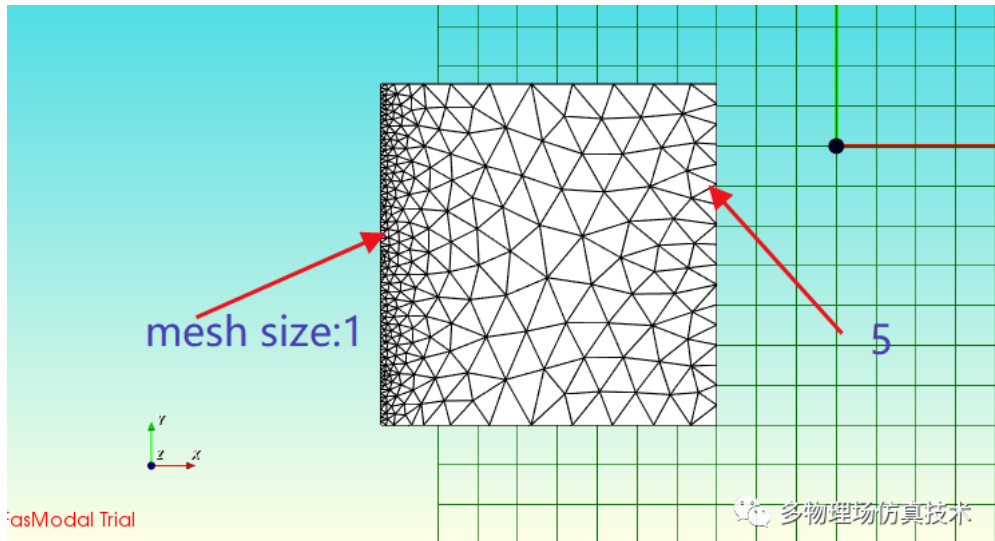
2. 局部网格尺寸

局部网格尺寸用来指定某些特定区域和几何上的网格尺寸，比如边界，激励，荷载，或者物理场梯度大的地方。局部网格尺寸的好处在于避免全局网格加密，在保证精度的前提下减少网格数量

3.增长率

网格增长率表明了相邻网格尺寸变化程度，是控制网格质量的一个重要指标。增长率为1表面网格尺寸没有变化。增长率一般控制在2，不会超过4，不同类型网格设置稍微有些偏差。

当我们设置的局部尺寸通常会小于全局尺寸，在不同尺寸网格之间存在一个过渡，这个过渡要尽量平滑，避免突变，如图。



FasModal中的局部网格尺寸设置

4.阶数控制

高阶网格需要增加新点，并且涉及到网格重编号。

5.弦高

用来指定曲线曲面离散的最大弧长与高度之比

6.跨度角

指定曲面生成的单元之间的面面角度(二面角))

7. 网格层数控制

在CFD，复合材料等领域，需要明确的设置边界上的网格层数，比如流体的边界层

8. 设置硬点 (fixed/hard point)

我们往往希望在某些特点的区域布置网格，但又不希望在几何上进行操作，可在网格中布置硬点，所谓的硬点即一定会出现在网格单元的节点列表里。

除了直接布点，还可以在边，面上有规律的设置点，比如在边上呈等比增加设点，也就是通常所说的布种子(seeding)。

9.网格加密

网格加密是在已有网格的基础上进行细化操作，涉及到网格修改，编辑，删除等操作，参考[深入理解数值计算网格\(8\)--自适应迭代网格](#)

2.网格数据结构

网格数据结构中除了基本的单元设计，还有一个非常重要的数据结构：

网格管理结构类

该数据结构中存储了网格的拓扑信息，还有相邻的拓扑连接信息，比如单元的相邻点线面和相邻单元，点的相邻单元，面的相邻单元，每个单元和点线面上的材料属性，边界荷载信息，历史记录，还有网格合并，加密，自由度计算等，是网格功能开发中的重点，后续在商业开发中详细介绍。

3.网格质量评价

网格质量分为**单个网格质量**和**整体网格质量**

单个网格质量控制包括：

最大角度

最小角度

最小边

最大边

Aspect ratio --最长边与最短边之比（不同类型单元有不同计算方法）

扭曲比

二维和三维，三角形和四边形，四面体和六面体分别有不同的评价指标

Aspect Ratio是比较通用的网格质量评价指标。

整体网格质量

包括数量，密度，单个网格质量的统计分类，与物理场梯度映射程度等等。这些都有比较成熟的统计方法。

4.网格策略

主要涉及到采用什么样的方法生成网格，这个一方面和模型几何有关，和仿真业务也紧密相关。

比如对于规则的长方体，可以采用映射法直接划分六面体，对于异形几何，只能采用自由划分四面体。一般只有通用有限元软件中会涉及到网格策略的选择，专业软件中会把网格策略设置好，只在高级选项中出现，一般情况无需用户干预。

6.理解高阶单元

“由于高阶数值计算复杂性，目前大部分商业软件网格都是线性单元（0阶）或者二次单元（一阶）。实际上高阶网格（也就是常说的P单元）在处理复杂几何上更具优势，适当的设置网格参数，只需更少数量的网格，无需迭代，就能获得更好的数值计算解，期待未来在高阶领域有更多实际应用。”

----[仿真软件十年回顾和展望\(整理版\)](#)

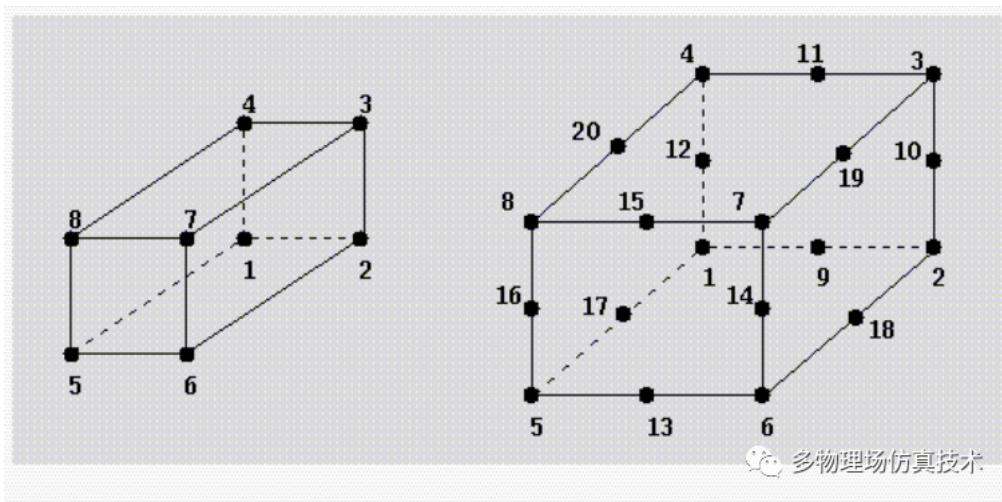
关于高阶单元，之前的文章有过很多介绍。当网格密度偏小，计算精度不够的时，可以通过两种方法提高精度，一种是加密网格（h单元方法），另一种是提升网格的阶次(p单元)，不再多描述。

下面主要介绍高阶单元的一些特点和实现

1.单元划分和生成

0阶单元，1阶单元的网格几何信息并没有改变，也就是说原来的点，边，面的位置没有发生改变。区别在于高阶单元的边或者面上点的增加。

对于网格划分来讲，难度并没有变化，区别只是在已有单元的边上再加一点。如果单元编号约定规则，把线性单元变成高阶单元在求解器阶段进行也可以。但考虑到模块解耦和功能独立性，一般还是把高阶网格的创建放在网格生成阶段。

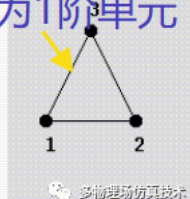


左为线性六面体单元，右为曲线六面体单元

在实际应用中，发现有些软件将线性单元（即三角形单元中仅有三个顶点）描述为1阶单元，有些描述为0阶单位，在沟通中造成了歧义。

为了避免歧义，统一将顶点仅在边两端的单元（三角形三个顶点）称为线性单元，将边中间有点的单元称为曲线单元。

有些软件定义为0阶单元，有些定义为1阶单元

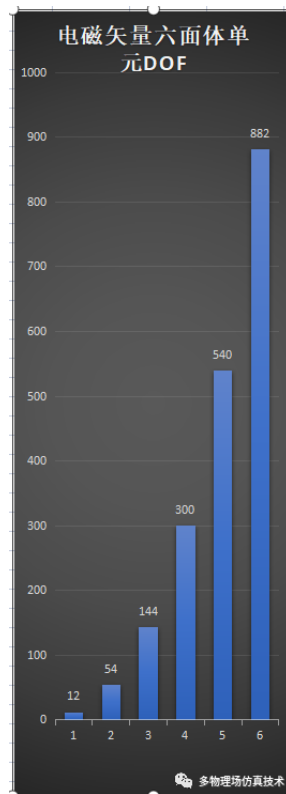


2. 高阶单元计算精度和计算资源

高阶单元之所以能提高精度，简单理解就是高阶单元使用的多项式变量次数更高，表达式更丰富，有能力更精准的反应单元内部物理量的变化。

大部分物理场的本构方程为二阶偏微分方程，或者可以转为二阶偏微分方程。高阶单元的基函数和形函数偏导后仍然能保证单元内部的非线性！

精度的提高的代价是计算量的快速增加，尤其是矩阵规模的增加



电磁六面体单元自由度数目随阶数变化

3.高阶单元实现

高阶单元实现有两种，一种是直接使用点插值的基函数方法，另一种是使用结构化组合基函数。

两种方法都基于线性单元，区别点在于第一种方法由单元上的一系列点来定义，高阶单元由增加插值点构成，因此在各个维度都有较好的独立性，且生成的刚度矩阵在预条件处理上有优势。多项式的每个表达式的系数有较好的物理含义，并且在边界上有较好的表达。不同阶数的单元有相同的统一表达式，代码更方便实现。

结构化组合基函数是由在低阶基函数的基础上乘以或加上新的函数。这样做的好处是在同一模型中可以使用不同阶的基函数，实现上灵活，但也增加了复杂度。

在实际研发中，更多的是采用第一种方法插值基函数的高阶单元。

混合单元

有限元等数值方法是在全局进行计算，所以可以在网格不容易生成的地方采用高阶单元，其它地方使用低阶一般单元，发挥各自的优点。

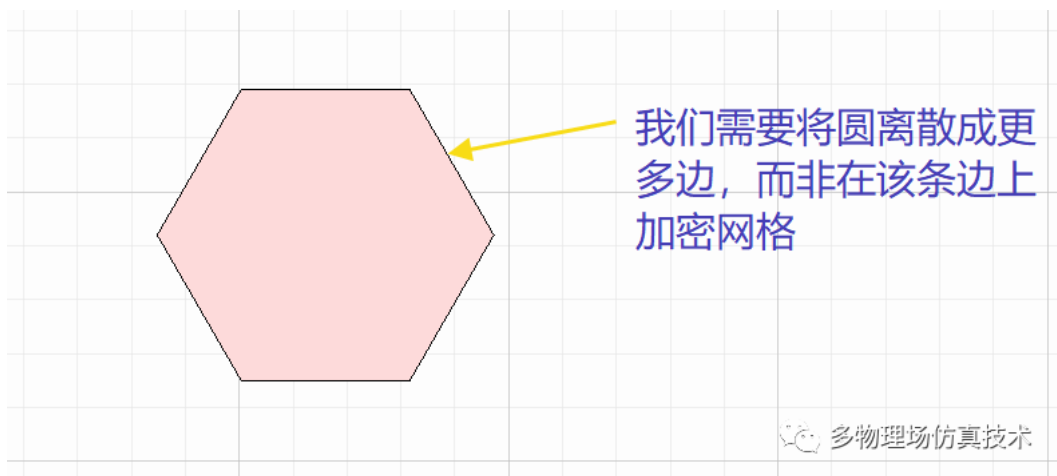
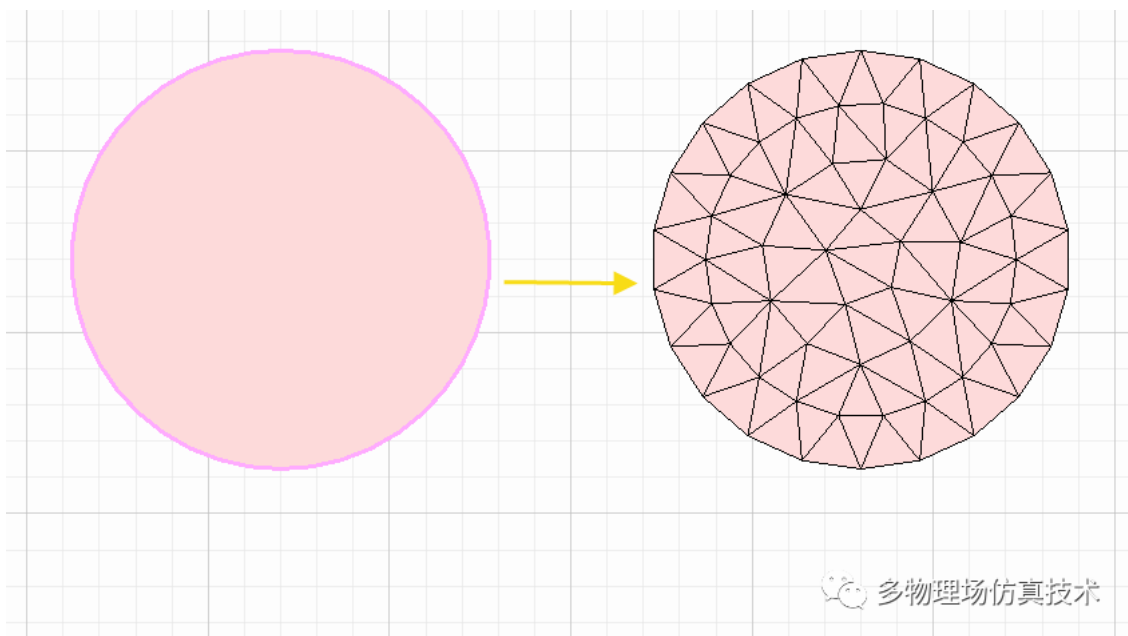
大部分有限元软件都支持高阶单元，在工程应用中，碰到网格划分困难，网格质量差，使用低阶单元收敛困难，且计算资源没有瓶颈的情况下可以尝试使用高阶单元。

7.几何与网格

我们知道网格的输入为几何数据，几何数据可以是参数化数据，BREP结构数据，或者其它任意形式定义的数据。

1.网格加密和几何

在划分网格的时候，需要将其离散化分成多段。如图，圆被离散成多条直线。如果在网格加密过程中，需要继续对圆弧进行加密时，没有原始几何信息，我们将只能加密在22条直线上，而无法将圆离散成更多的边。实际应用中，对圆弧网格尺寸加密是要求将其离散的更密集，而非在离散的边上加密。



2. 共形网格与几何

在三维接触碰撞，CFD，复合材料，多物理场耦合领域分析中，需要将不同属性，不同材料对象放在一起分析。在网格上需要将之前对象属性保留。这就会碰到一个难题：如何处理不同对象接触部分的网格。

通常做法有以下几种：

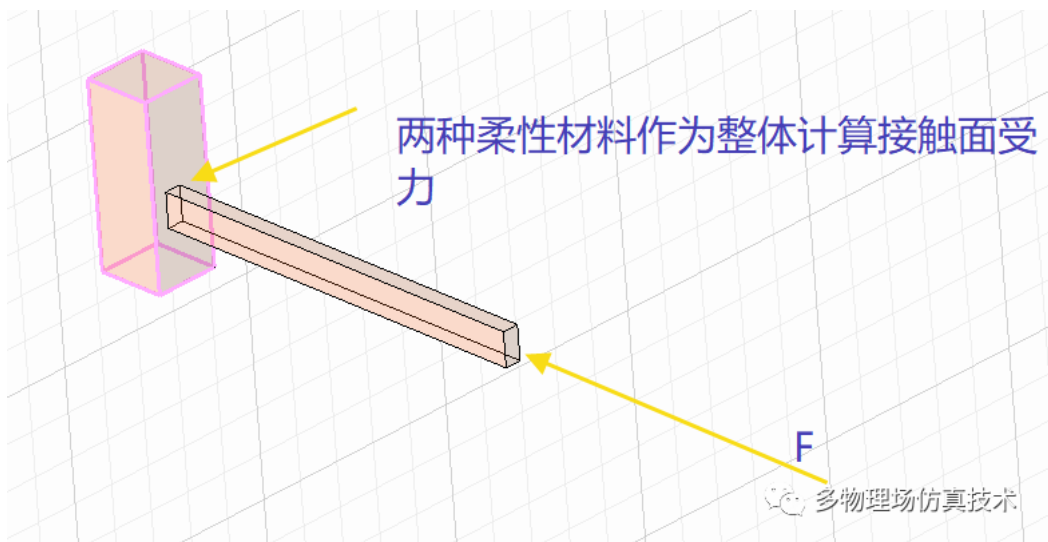
- 1.将各个对象作为独立个体，将其中一个作为主要分析对象，其它对象简化；
- 2.各个对象分别划分网格，设置不同属性，将可能接触部分的网格标识出，让求解器去处理；
- 3.在可能接触部分进行网格平滑，保证点和边能完全重合，但不保证接触对象完全一对一；
- 4.划分点，边，面完全一对一的网格，也就是通常所说的共形网格

(conformal mesh)

其中第四点共形网格由于保留了物体原有属性，且能把所有对象在同一整体刚度矩阵中分析而成为大多数仿真软件的选择。

在几何上，设置不同的材料，属性是很容易的，但在共形网格划分上，由于通常是先生成面网格，在面网格基础上再生成体网格，造成了各种属性的传递困难。

以材料为例，当两种不同材料物体接触一起分析时，划分面网格阶段，我们需要区分每个面在哪种材料上，按照一般规则，我们可以定义面的法向量的方向为材料的方向，这也就需要在几何中非常明确规定实体上的面的方向，而不能毫无规律！对于接触部分，需要在几何中明确计算出具体位置，然后传递给网格引擎。



3. 几何清理

做一个简单类比，求解器是发动机，网格是汽油，几何则是原油。网格生成则是炼油的过程，高质量的原油可以炼出好的汽油。

为了生成高质量的网格，干净正确的几何输入是必要前提，好的几何也有助于后期网格的调整，编辑和优化。

相信很多工程师有清理几何的痛苦经历。设计生产的CAD几何模型往往不适用直接仿真，比如不必要的文字，几何不连续，细小边，细缝，物体干涉，倒角，圆孔，自由对象，重复对象，不同格式之间的转换带来的几何拓扑错误，容差错误等等。拓扑错误，几何不封闭，方向错误，连接顺序相反也是常见的几何错误。

4. 虚拟拓扑

虚拟拓扑是针对几何到网格过程中繁琐的操作进行的一种简化操作。

场景1：需要在实体的一个面上某一小块区域A而非整个面进行边界或荷载设置，通常做法需要创建新的面几何把这一块区域覆盖。另外一种做法可以将区域A设置为虚拟拓扑，直接传给网格引擎，让网格引擎单独处理。

场景2：需要在很多个小的面上进行荷载和边界设置，常规做法是需要一个个细小的面选择，然后进行设置，可以将这些面设置成一组，作为一个单独面使用，大大简化操作。

针对虚拟拓扑，可以设置过滤条件，让很多手工操作半自动化或者全自动化，尤其是大模型，能减轻工程师的负担，也减少操作失误的几率。

5. 几何曲线曲面处理

曲线曲面的网格划分

在几何中，曲线曲面通常分为三种：参数，显式和隐式。参数形式往往有严格的方程定义；显式几何明确给出点的位置或面片；隐式往往以表达式的显示出现。

仍然以圆为例

参数形式为：

$$x=a+r*\cos\theta$$

$$y=b+r*\sin\theta$$

参数表达式很容易满足连续性要求，满足1,2阶偏导要求，在网格端很方便拿到参数。

我们可以显式给出圆周上16个点，点顺序相连，给出圆的近似表达。

显式几何的好处是网格无需做过多处理，缺点是需要明确几何的正确性，比如16段圆弧才能满足仿真精度，但几何只给出8段，那么在网格端再如何处理也是无法达到精度要求。

隐式形式：类似公式 $x^2+y^2=A$

隐式表达的好处是给出任意一点坐标，我们能利用表达式方便判断出它的相对位置。隐式几何精准将几何信息传递给网格，对网格处理上也提出了更多要求，在曲线曲面离散上，要取到几何的本身属性以及控制参数，另外还要考虑网格参数，如果出现冲突，要选择合适的算法处理，实际中只有在特定场合才使用。

6. BREP结构和网格

主流的三维几何内核表达为BREP结构，利用这种结构，网格很容易拿到BREP结构中的几何和拓扑信息，方便进行后续网格划分。

为了显示需要，一般几何内核通常也具有三角化的功能，几何离散成的三角形称之为“面片”，非网格，大部分显示引擎底层都使用三角形来渲染对象，因此2维、3维几何都需要三角化（称之为“面片化”），比如一个长方体共有6个面，每个面需要离散成两个三角形，总共12个三角形。这12个三角形我们称之为12个“面片”，而非网格！一般情况下“面片”质量很差。“面片”有两个功能：一是用来做显示渲染数据；二是可以作为网格划分的输入数据，用来生成面网格。

也就是说，网格划分既可以直接拿几何的数据结构，也可以拿三角化之后的数据，可以根据实际需要做选择。

7. 变形网格(morphing)+高阶单元

morphing技术是网格划分中的一种常见方法。即初次划分网格后，几何发生变化，网格也跟随发生变化，但网格只是点坐标跟随变动，而拓扑(连接性)不发生变化。morphing在参数化几何设计中非常有用，由于不需要重新划分网格，提高了设计效率。借助于高阶网格单元，可以在几何设计阶段快速完成高精度仿真，也是generative design的基础性技术之一。

8.自适应迭代网格

在前处理网格划分中，如果网格数量偏少，在物理场变化大的地方，网格内部不足以表达这种变化，从而导致计算偏差。类似于用多边形模拟圆形，线段数量越多，越接近圆形。

网格加密通常有以下几种标准：

1. 根据几何特征，比如在曲面，曲线，孔边等加密。这种加密依赖于几何拓扑信息，但并不准确，比如在尖角处网格加密反而得出错误的结果。
2. 根据业务特点，比如流体在边界层，电磁仿真在金属部分需要加密，在初次划分网格的时候就可以进行设置；
3. 根据两次仿真所得物理参数结果，比如在应力/压力/速度/温度变化梯度大的地方；
4. 根据两次仿真业务数据进行加工，比如HFSS根据S参数误差，能量误差，CFD中根据压力，速度误差。

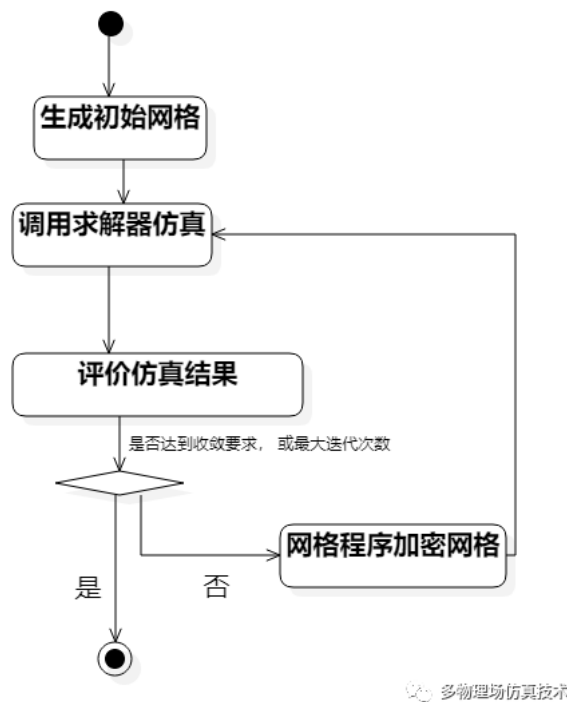
1.加密策略

加密策略涉及到什么时候加密，如何加密，加密效果如何评价等等。

按照加密的阶段，可以分为两种，一种是在仿真之前加密，一种是在仿真计算后加密。

仿真前加密，是利用对象已有的特性，包括几何特性和仿真属性，几何特性比如孔边，大曲率曲线曲面，仿真属性比如接触单元，有电流经过的金属，明确确定物理场会发生突变的单元，确定的荷载，边界条件位置。仿真前加密一般基于已有仿真经验，加密单元的位置比较准确，但是仿真结果的准确度很难确定。

仿真后加密就是一般性自适应网格加密，UML活动图如下：



多物理场仿真技术

其中如何“评价仿真结果”是很重要的一环，一般的做法是“根据两次仿真所得物理参数结果，比如在应力/压力/速度/温度变化梯度大的地方”，也有根据单元内部数值计算误差的统计误差作为评估标准。每种物理场的评价指标和方法都不尽相同，但总的原则都是相似的。

2. 加密算法和参数

在网格加密流程中，求解器会给出需要加密的单元。这就涉及到两个问题：

1. 如何使用加密算法
2. 加密尺寸如何确定

给出需要加密单元，不考虑效率，其实最理想的方法是重新划分网格，在给出需要加密尺寸的地方设置局部网格参数，但这种方法比较耗时，而且如果加密单元少的话，比较浪费资源。

常用的做法是在已有网格的基础上进行加密，直观的方法是将原有网格拆分成更小。这种做法理论可行，但实际应用中会碰到很多问题，首先是网格质量无法保证，以三角形为例，一个等边三角形，正中间加点分成三个三角形，每个三角形的最大角度成为120度的钝角三角形；其次网格数量很难控制，二维三角形单元必须分为三个，而三维四面体则必须分为9个，如果加密网格尺寸只要加密到原尺寸的0.8。这种直接拆分法无法满足需求；最后直接拆分网格，导致相邻单元的尺寸梯度突变，反而不符合物理场的平滑过渡的要求。

加密尺寸可以由求解器给出，如果求解器只给出了加密单元，而没有加密尺寸。就意味着，加密尺寸需要在网格自适应迭代过程探索，增加了网格加密和求解迭代的次数。

在给定加密单元和尺寸的前提下，通用的方法还是类似波前法，将所在区域挖空，然后根据加密尺寸局部参数要求，重新生成新的网格，新网格生成后，再对周围网格进行优化微调。在该过程中，涉及到了网格的删除，编辑，再生成，再编号等一系列操作，开发难度要高于初始生成网格。

3. 收敛问题

收敛问题涉及网格划分中经常碰到另外一个问题：

网格是否越密越好？

答案是否定的

网格不可能无限度的加密，因为过度加密即使没有加密上的误差，也会引起数值计算上的误差。比如建筑的仿真尺度以米为单位，无限加密到毫米尺度时，如果原始单位为米，则会出现数值计算上的误差，也就是说网格加密并不是因为物理场本身的变化，而是数值误差引起的变化。

如果网格已经满足了仿真精度的要求，再加密网格并不会增加仿真结果的精度，除了徒增加计算资源外，还会造成计算收敛结果的震荡。

网格收敛同样也要符合模型的计算规则，比如针对结构有限元中的沙漏和自锁现象，网格加密的力度要比较激进，而非普通的渐进式的加密。

网格加密的收敛是比较适合用AI算法进行，因为实际应用中，每种产品的结构基本上不会有太大的变化，这就使得网格的参数能较好的进行统计计算；网格的加密也是迭代生成，天然适合AI算法的训练；当所有网格参数训练完善时，自适应加密网格也就不需要了，直接在初始网格阶段敲定所有参数。

4.商业软件现状

很早以前网格的划分依赖于工程师的经验，很多时候需要在算出结果后重新划分网格，来回折腾。现如今自适应加密网格已经成为商业仿真软件的标配，如果你使用的软件还没有使用到自适应网格加密，可以试试其它软件。

9.商业开发

了解公众号的朋友都知道，本公众号既不是流量自媒体，也不做一般的科普和工业软件使用教程。公众号的所有文章都是围绕工业软件开发底层技术而谈！

网格生成作为工业仿真软件的一项重要的基础性功能，必然也是研发的重点和难点。在前期介绍网格的基础上，本文就简单聊一聊网格相关功能的商业开发。

有朋友可能会有疑问，商业开发和一般的开发区别在哪里？

其实商业开发的难点主要在于解决实际工程问题：一方面基本的功能都要有，另一方面要能满足工程中各种稀奇古怪的需求，有些可能是反常识的工程问题。在一般功能的基础上，对网格生成的性能，稳定性，兼容性都有更高的要求。

比如我们常用的几何内核为ACIS和Parasolid，网格引擎需要能直接读入ACIS和Parasolid的数据，避免二次导入的数据损失；现在的模型越来越大，设计越来越复杂，网格要求有更精准的参数控制来处理大模型，满足性能要求，生成符合实际需求的网格；针对逆向工程和AI建模等新趋势，网格模块要能处理扫描出的大量无用数据和脏数据，有能力进行数据清洗等预处理工作；此外网格智能加密和稀疏，支持混合单元，网格拆分组合，网格修复，虚拟拓扑，并行网格划分等也一直是应用领域的刚需。

从功能需求看，网格的开发分为两部分：

1. 网格引擎开发；
2. 业务应用开发；

网格引擎提供底层网格划分技术，包括标准几何导入识别，实现网格划分算法，网格优化，生成结果网格数据，导出标准中间网格格式等等，是基础性功能研发。目前这块整体看开发资源相对较少，开源的比如getgen，tetgen，gms，CGAL都有提供，可以作为学习和参考的资料。综合性的功能比较全面的商业组件选择余地小。

业务应用开发在网格引擎的基础上，实现业务的实际需求，比如自动设置合理材料，边界等信息，最终生成求解器需要的网格文件。这块直接和业务挂钩，是大部分网格开发的重点。

很多专业软件因为对网格的需求比较集中和明确，因此开发往往同时包括了网格引擎和应用业务开发。这也是未来仿真软件开发的趋势，即根据自己业务需求开发网格功能，掌握和开发底层网格生成算法！

本来打算在商业开发中讲讲网格引擎的设计和实现，以及业务应用开发。发现内容太多，泛泛讲没有深度，也没有太多指导意义，后续有空再聊。

从十多年的工业软件开发经历看，网格作为基础性功能，技术更新十分缓慢。更新主要集中在性能比如多进程，分布式计算，几何接口扩充，实际仿真业务需求等。可以预见，未来也不会有太多更新。相比求解器，几何内核，网格功能相对独立，且内容比较集中，没有太高的瓶颈和工作量，更多是算法上的实现和完善（注意：只是相比几何内核和求解器）

从国外大环境看，各大厂商也都在加紧布局工业软件的底层核心业务，换句话说，以后在全球都很难买到底层应用模块，甚至使用可能都会比较困难。

目前国内工业软件发展如火如荼，想在这块有所作为，网格功能要早早介入底层开发，进行长期技术上的积累。

相比之前的网格[入门介绍](#)，“深入理解数值计算机网格”系列介绍了更多工业仿真软件中的网格细节，但是相比网格功能算法的具体实现，实际应用，仍然只是皮毛。后续有空会更多介绍网格引擎的设计，算法实现，网格和业务的结合以及在工业实际中的应用。

[一篇文章入门网格划分](#)

[深入解析HFSS自适应网格加密策略](#)

[如何构建世界级的中国工业仿真软件](#)

阅读: null

在看: null