

基于黎曼度量张量的各向异性网格划分算法研究

严科科, 陈 兴*

(宁波大学 机械工程与力学学院, 浙江 宁波 315211)

摘要: 由于实际中某些复杂性工程问题的解具有各向异性的特点, 为采用更少的网格单元数及更好的单元质量来进行有限元分析, 以实现高效求解, 各向异性剖分单元则是一种有效的前处理技术. 因此为生成高质量各向异性网格, 首先在给定黎曼度量的基础上形成各向异性背景网格, 然后通过各向异性 Delaunay 原则进行边交换, 再基于力平衡实现节点的光滑平顺, 由标准化面积和标准化边长规定节点的添加与删除, 以及节点近似投影的边界约束, 得到一个与具有方向性问题相匹配的网格. 最后通过 3 个实例验证给出的各向异性网格划分算法的可行性.

关键词: MATLAB; 网格生成; 各向异性; Delaunay

中图分类号: O186.12

文献标志码: A

文章编号: 1001-5132 (2020) 04-0023-07

有限元是现代科学中解决复杂问题的有效方式, 通常由建立几何模型和网格划分(前处理)、计算求解、结果处理和评定(后处理)三部分组成^[1]. 各阶段所占的时间比一般为 40%~60%、5%~10%、30%~50%. 其中网格划分是前处理中极其关键的一步, 可以较大程度地影响到最终有限元计算结果的准确性和有效性. 对于某些工程问题, 例如计算具有方向性的高分子黏性流体流动问题时, 使用各向异性网格可以更高效地解决问题.

已有学者通过各种方式来研究各向异性网格生成算法. Simpson^[2]提出基于各向同性网格进行坐标变换的各向异性非结构网格生成算法, 但该算法只适用于常数度量的情形且对边界需要另外处理. Bossen 等^[3]提出一种柔顺网格生成方法, 在插点、删点的迭代过程中进行网格光滑, 其结点光滑函数是基于节点相互作用力的平衡而建立的; Du 等^[4]提出 ACVT(Anisotropic Centroidal Voronoi Tessellation)划分的各向异性网格光滑算法, 该算法基于对偶剖分的原理生成了各向异性 Delaunay 三角化网格. 关振群等^[5]将 AFT(Advancing Front Technology)和二分法相结合, 生成二维各向异性网格. 还有学者为表达各向异性网格的特性, 引用

黎曼度量张量场(Riemannian Metric Tensor Field)来描述网格节点的方向场和尺寸场信息, 例如 Borouchaki 等^[6]给出了在黎曼空间中实现二维平面区域的各向异性网格生成算法, 具体实现时采用 AFT 产生新节点, 在黎曼空间中用 Delaunay 原则进行边交换, 最终形成各向异性网格. 粟一凡等^[7]提出在各向异性背景网格中通过力平衡函数优化节点来生成各向异性网格.

本文拟通过引入黎曼度量, 结合各向异性 Delaunay 原则, 基于力平衡的节点光滑平顺和节点的近似投影边界约束在计算软件包 MATLAB 中生成高质量的各向异性三角化网格.

1 各向异性基础

各向同性网格和各向异性网格都是相对于度量空间的. 如定义在欧氏空间度量场中的各向同性网格, 网格节点沿任意方向的尺寸值相同; 但对于欧氏空间度量场中的各向异性网格而言, 网格节点沿特征向量的尺寸值是不同的; 而在黎曼度量张量场中, 各向异性网格节点沿任意方向的尺寸值都是一样的. 因此三角形网格划分算法可通过引入黎曼度量以实现各向异性的划分^[8-10].

收稿日期: 2020-01-20.

宁波大学学报(理工版)网址: <http://journalg.nbu.edu.cn/>

基金项目: 材料成形与模具技术国家重点实验室开放项目(P2016-21).

第一作者: 严科科(1994-), 男, 浙江宁波人, 在读硕士研究生, 主要研究方向: 网格划分算法. E-mail: 837603690@qq.com

*通信作者: 陈兴(1963-), 男, 湖南茶陵人, 副教授, 主要研究方向: 材料成型 CAD/CAE. E-mail: chenxing@nbu.edu.cn

各向异性度量的定义为给定二维区域 Ω 上存在黎曼度量 M , 其为对称正定矩阵, 将其进行正交分解如下:

$$M = E^T \Lambda E = [e_1, e_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [e_1, e_2]^T = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} h_1^{-2} & 0 \\ 0 & h_2^{-2} \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}, \quad (1)$$

式中: Λ 为尺寸场; λ_1 和 λ_2 为特征值; h_1, h_2 是在向量方向的期望尺寸; E 为方向场; e_1, e_2 为单位正交化特征向量; θ 为旋转角度函数(图 1).

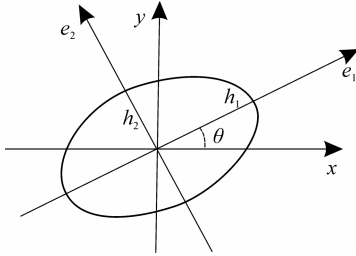


图 1 黎曼度量矩阵控制下的椭圆

假如 Ω 上有两点 p 和 q 距离足够小, 则两点间的度量变化细微, 则可以求得两点在给定度量场中的近似距离:

$$L_M(p, q) = \sqrt{\vec{pq}^T M_{\text{avg}} \vec{pq}}, \quad (2)$$

式中: $M_{\text{avg}} = (M(p) + M(q))/2$ 是两点度量平均值.

同样, 可以给出在给定度量场中的近似三角形面积公式:

$$S_M(\Delta_{abc}) = \sqrt{\det(M_{\text{avg}}(\Delta_{abc}))} S(\Delta_{abc}) = \frac{1}{2} \sqrt{\det(M_{\text{avg}}(\Delta_{abc}))} (b-a) \times (c-a), \quad (3)$$

式中:

$$M_{\text{avg}}(\Delta_{abc}) = \frac{M(a) + M(b) + M(c)}{3}, \quad S(\Delta_{abc}) \text{ 是}$$

Δ_{abc} 在 Euclid 的面积.

各向异性网格质量公式为:

$$q_i = \frac{4\sqrt{3}S_M(t)}{\sum_{i=1,2,3} (L_M(t_i))},$$

式中: $S_M(t)$ 由式(3)给出; $L_M(t_i)$ 由式(2)给出的三角形边长. 当 $q_i=1$ 时, 即三角形在当前度量下为等边三角形. 为比较网格优劣, 使用平均网格质量

q_{avg} 作为判定依据, 且设网格总数为 N ,

$$q_{\text{avg}} = \frac{1}{N} \sum_{i=1}^N q_i. \quad (4)$$

同时将黎曼度量引入到 Delaunay 原则中实现边翻转, 推出各向异性 Delaunay 原则^[11]. 如图 2 所示, 三角形 Δ_{abc} 和 Δ_{acd} 组成一个四边形 $abcd$, 在 $abcd$ 上定义局部常度量:

$$M_{\text{avg}} = \frac{M(a) + M(b) + M(c) + M(d)}{4}.$$

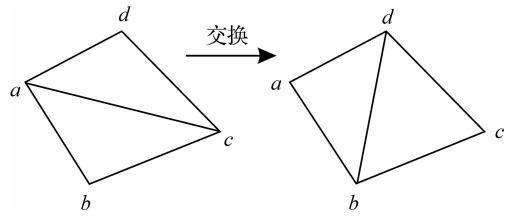


图 2 黎曼度量场中的边翻转

那么各向异性 Delaunay 原则为:

$$\left(\vec{bc} \times \vec{ba} \right) \vec{da}^T M_{\text{avg}} \vec{dc} + \vec{bc}^T M_{\text{avg}} \vec{ba} \left(\vec{da} \times \vec{dc} \right) < 0. \quad (5)$$

当式(5)成立, 即在黎曼度量场中的 $\angle abc + \angle adc > \pi$ 时, 交换边 ac 为 bd (图 2).

2 各向异性网格划分算法

为实现二维变度量各向异性网格划分, 首先依据黎曼度量和设定的背景网格尺寸值, 在给定隐函数的区域中生成具有各向异性特征的网格; 再由节点间的力平衡来优化结点位置, 之后判定是否需要增加或删除节点, 并以近似投影来实现边界约束; 最终生成优质的各向异性三角网格.

主要算法实现框架如下:

(1)在给定区域内创建均匀的节点分布, 隔行交错移动, 然后将边界外的节点删除.

(2)对剩余节点进行各向异性 Delaunay 三角划分, 并建立各个三角形对应邻居的数据结构.

(3)进行主循环. ①以各向异性 Delaunay 原则进行边翻转算法, 同时更新邻接结构. ②对划分后的三角形重心进行判定, 移除在边界外的三角形, 同时更新邻接结构. ③节点的光滑平顺. (a)计算出三角形各边在黎曼度量张量场中边长; (b)以力平衡函数计算出各边所受力的大小; (c)将各边的力拟

合到各个节点上, 计算出各个节点的合力, 得出新节点; (d) 判定原节点与新节点所形成线段是否与原网格线段相交. 如果不相交, 则移动到新节点; 如果相交且相交于网格边界线段, 则移动到交点处; 如果相交, 则不移动. ④判定是否添加或删除节点, 则同时更新邻接结构. ⑤判定各个节点是否在边界内, 如果在边界外, 则将节点映射回边界上.

(4) 终止循环, 判定条件由所有节点的最大移动距离是否小于设定值或循环次数大于设定循环次数.

其中边翻转、节点的光顺、插入和删除以及边界外节点的移动为核心算法, 分别叙述如下.

2.1 边翻转算法

为实现更高的性能和稳健性, 可通过改变局部网格连接来控制三角网格划分质量. 在主循环迭代中, 更新网格节点之间的连接来维持具有高质量的网格^[12]. 但是由于大多数三角形网格已经具有较高质量, 重新划分网格时仅需要修改其中一些网格元素. 边翻转算法可实现局部网格的更新, 在实现边翻转算法时, 需要添加一个数据结构, 它提供了指向每个三角形邻接三角形的指针, 每当重新划分网格时都必须修改它们. 网格内部边翻转算法具体如下:

(1) 从初始三角形网格中提取出内部边, 组成队列.

(2) 进行队列循环. ①从网格单元中提取队列第一条边对应的三角形组成四边形, 并删除第一条边; ②当四边形不符合各向异性 Delaunay 原则时, 将新边插入队列.

(3) 队列为空, 终止循环.

边翻转算法的实施不仅可以提高网格质量, 也保持了其余网格的完整. 同时由于这种局部网格划分不必进行整体网格划分, 因此显著提高了网格划分效率. 局部网格划分的另一个优点是可以保留初始网格的拓扑信息.

2.2 节点的光顺

为提高网格质量, 极其有必要研究网格节点的光顺算法. 例如, 用节点相邻点组成多边形的重心来代替节点的 Laplace 光滑算法^[13], 通过 Lloyd 迭代, 用 Voronoi 区域的重心来代替相应节点的

CVT 光滑^[14-15]; 通过构建理想单元的矩阵来获得节点优化位置的 TMOP(Target-Matrix Optimization Paradigm)算法^[16]. 本节采用基于力平衡的节点优化方法^[3,17], 该方法通过控制节点间的距离, 给出了光顺优化函数, 并得出较优网络节点的位置.

如将两点 p_1 和 p_2 之间的标准化距离 $d(p_1, p_2)$ 定义为:

$$d(p_1, p_2) = L(p_1, p_2) / r, \quad (6)$$

式中: r 是期望边长. 最终目标为划分出标准化边长为 1 的网格.

在给定的标准化距离定义下, 光滑算法可以用以下方程表示:

$$p_i' = p_i + \alpha_i \sum_j f(d(p_i, p_j)) u_{ij}, \quad (7)$$

式中: p_i 为节点 i 的坐标; u_{ij} 为单位向量 $(p_i - p_j) / d(p_i, p_j)$; j 为节点 i 的邻域节点组成的点集, f 是平滑函数; α_i 为常数, 通常被设置为 0.2.

如图 3(a)所示, 在 Laplacian 平滑中, $\alpha_i = 1/n$, 而 n 为与节点 i 相交的节点数之和, 并且 $f(d) = -d$. 如图 3(b)所示, Persson 在 Distmesh 中提出基于类似弹簧的力平衡函数 $f(d) = \max(1 - d, 0)$.

如图 3(c)所示, Lennard-Jones 描述一种可表现吸引/排斥的力平衡函数 $f(d) = d^{-13} - d^{-7}$. 但对于网格生成而言, 该模型则存在数值不稳定性, 因为 $f(0^+) \mapsto \infty$, 因此, 有学者提出了 Lennard-Jones 力平衡函数的几种变体. 例如 Shimada 提出的力平衡函数(图 3(d)):

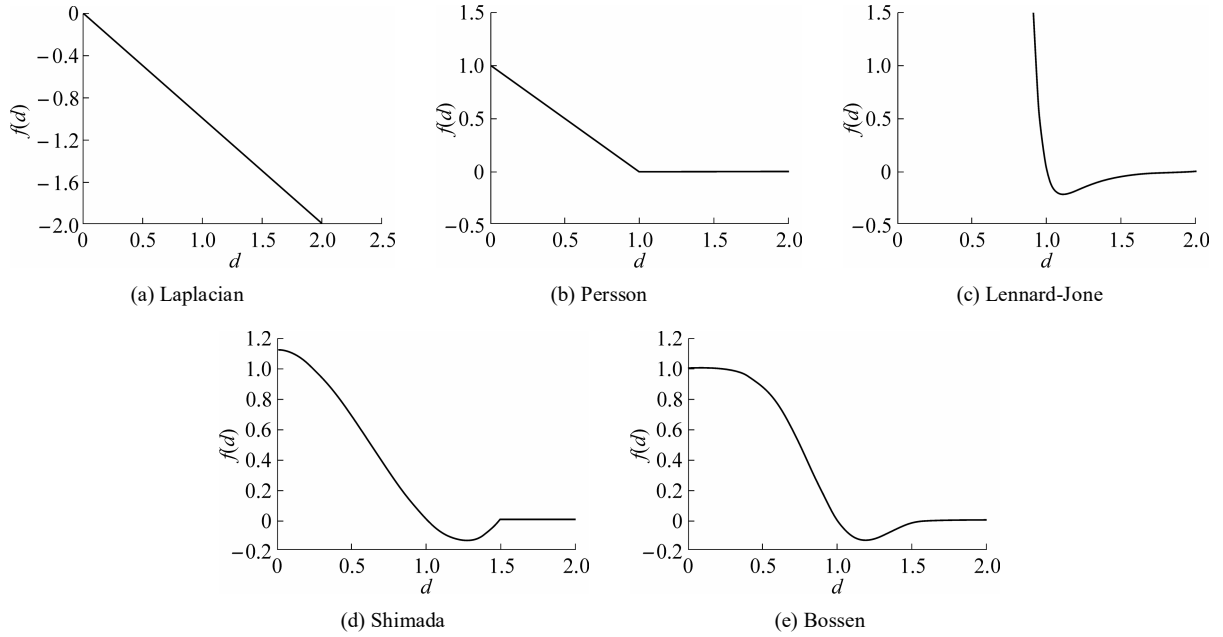
$$f(d) = \begin{cases} k(1.25d^3 - 2.375d^2 + 1.125), & 0 \leq d \leq 1.5, \\ 0, & 1.5 < d. \end{cases} \quad (8)$$

如图 3(e)所示, Bossen 提出力平衡函数 $f(d) = (1 - d^4) \times \exp(-d^4)$, 该函数描述了节点距离其他节点太近 ($d < 1$), 则会排斥, 如果距离太远 ($d > 1$), 则会吸引.

从上述几种力平衡函数可以看出, Bossen 的力平衡函数具有吸引/排斥的平滑函数, 且具有更好的光顺结果. 因此, 本文采用 Bossen 力平衡函数.

2.3 节点插入和删除

当网格局部稀疏时, 需插入节点; 而当网格过于密集时, 需删除节点. 文中使用标准化各向异性边长和标准化各向异性面积来进行判定节点的插

图3 5种力平衡函数 $f(d)$

删操作^[17]. 标准化各向异性边长可由式(6)进行描述, 而标准化各向异性面积公式可表示为:

$$|S(\Delta_{abc})| = \frac{S(\Delta_{abc})}{r^2},$$

式中: r 为期望边长.

具体节点插删操作算法如下:

(1)判断是否需要删除节点. ①边界上节点判定标准: 其相邻的2个边界边的标准化各向异性长度为 d_1 和 d_2 , 若 $(d_1^2 + d_2^2)/2 < 1$, 则删除该节点. ②内部节点判定标准: 其相邻的 n 个三角形标准化各向异性面积为 $S_i(p) (i=1, \dots, n)$, $\sum_{i=1}^n S_i(p)/n < \sqrt{3}/4 \times 0.8$, 则删除该节点.

(2)当节点不需要删除后, 判断所有与其连接的边上是否需插入节点. 判定标准如下: ①边界边判定标准: 其标准化各向异性长度 d_1 , 若 $d_1^2/2 > 1$, 则标记该边. ②内部边判定标准: 其相邻的2个三角形的标准化面积为 S_1 和 S_2 , 若 $0.8(S_1 + S_2) > 1$, 则标记该边.

如果存在标记边, 插入最长边的中点.

2.4 节点的近似投影

由于力平衡的作用可能会使节点移出所要求的几何边界外, 如果边界偏差值过大, 会造成几何图形与所求模型存在较大差距. 如果节点在几何边界外, 需要将节点重新投影回边界上. Distmesh

采用数值微分求解隐函数的梯度来投影节点^[17].

本文则通过近似隐函数, 将边界外的节点投影到边界中.

若定义一个节点 p 和隐函数 $\varphi(p)$, 为将节点投影到边界上, 若存在矫正值 Δp , 满足以下条件:

$$\begin{cases} \varphi(p + \Delta p) = 0, \\ \Delta p \parallel \nabla \varphi(p + \Delta p), \end{cases} \quad (9)$$

即矫正值 Δp 与隐函数中节点 $(p + \Delta p)$ 的梯度平行. 若为式(9)添加附加参数 t , 即采用拉格朗日乘数来改写公式, 从而得到:

$$\begin{cases} \varphi(p + \Delta p) = 0, \\ \Delta p + t \nabla \varphi(p + \Delta p) = 0. \end{cases} \quad (10)$$

2.4.1 一阶近似投影

通过在节点 p 处将截断的泰勒展开, 替换隐函数 φ 来导出一阶近似方程:

$$\varphi(p + \Delta p) \approx \varphi + \nabla \varphi \times \Delta p. \quad (11)$$

假设式(10)中节点 $(p + \Delta p)$ 的隐函数梯度 $\nabla \varphi \cdot (p + \Delta p)$ 和包含节点 p 的近似隐函数梯度一致, 则式(10)可变成:

$$\Delta p + t \nabla \varphi = 0. \quad (12)$$

将其代入式(11), 可得:

$$\varphi - t \nabla \varphi \times \nabla \varphi = 0 \Rightarrow t = \varphi / (|\nabla \varphi|^2). \quad (13)$$

同时, 可以求出节点 p 的位移为:

$$\Delta p = (\varphi / (|\nabla \varphi|^2)) \times \nabla \varphi. \quad (14)$$

2.4.2 二阶近似投影

通过截断更多隐函数 φ 的泰勒展开式来推导出近似投影公式, 对于节点 $p(x, y)$ 和它的位移 $\Delta p(\Delta x, \Delta y)$, 可以得到:

$$\begin{aligned} \varphi(x + \Delta x, y + \Delta y) \approx & \varphi + \Delta x \varphi_x + \Delta y \varphi_y + \\ & \frac{1}{2} \Delta x^2 \varphi_{xx} + \Delta x \Delta y \varphi_{xy} + \frac{1}{2} \Delta y^2 \varphi_{yy}. \end{aligned} \quad (15)$$

假设隐函数 φ 和其导数同包含节点 $p(x, y)$ 的近似隐函数一样, 那么由式(10)可以得到:

$$\begin{cases} \Delta x + t(\varphi_x + \Delta x \varphi_{xx} + \Delta y \varphi_{xy}) = 0, \\ \Delta y + t(\varphi_y + \Delta x \varphi_{xy} + \Delta y \varphi_{yy}) = 0. \end{cases} \quad (16)$$

解得

$$\begin{cases} \Delta x = \frac{(\varphi_y \varphi_{xy} - \varphi_x \varphi_{yy})t^2 - \varphi_x t}{(\varphi_{xx} \varphi_{yy} - \varphi_{xy}^2)t^2 + (\varphi_{xx} + \varphi_{yy})t + 1}, \\ \Delta y = \frac{(\varphi_x \varphi_{xy} - \varphi_y \varphi_{xx})t^2 - \varphi_y t}{(\varphi_{xx} \varphi_{yy} - \varphi_{xy}^2)t^2 + (\varphi_{xx} + \varphi_{yy})t + 1}. \end{cases} \quad (17)$$

将其导入式(16), 可以简化得到 t 的四阶多项式:

$$p_4 t^4 + p_3 t^3 + p_2 t^2 + p_1 t + p_0 = 0, \quad (18)$$

其中,

$$\begin{aligned} p_0 &= \varphi, \\ p_1 &= 2\varphi(\varphi_{xx} + \varphi_{yy}) - \varphi_x^2 - \varphi_y^2, \\ p_2 &= \varphi \varphi_{xx}^2 + \varphi \varphi_{yy}^2 + 4\varphi \varphi_{xx} \varphi_{yy} - 2\varphi_y^2 \varphi_{xx} - \\ & \quad 2\varphi_x^2 \varphi_{yy} - 2\varphi \varphi_{xy}^2 - \frac{1}{2} \varphi_y^2 \varphi_{yy} - \frac{1}{2} \varphi_x^2 \varphi_{xx} + \\ & \quad 3\varphi_x \varphi_{xy} \varphi_y, \\ p_3 &= -\varphi_x^2 \varphi_{yy}^2 - 2\varphi \varphi_{xy}^2 \varphi_{xx} - \varphi_y^2 \varphi_{yy} \varphi_{xx} + \\ & \quad 2\varphi_x \varphi_{xx} \varphi_{xy} \varphi_y - 2\varphi \varphi_{xy}^2 \varphi_{yy} - \varphi_y^2 \varphi_{xx}^2 + \\ & \quad 2\varphi \varphi_{yy}^2 \varphi_{xx} - \varphi_x^2 \varphi_{yy} \varphi_{xx} + 2\varphi_x \varphi_{xy} \varphi_{yy} \varphi_y + \\ & \quad 2\varphi \varphi_{xx}^2 \varphi_{yy}, \\ p_4 &= -\varphi_x \varphi_{xy}^3 \varphi_y + \varphi \varphi_{yy}^2 \varphi_{xx}^2 + \varphi \varphi_{xy}^4 - 2\varphi \varphi_{xy}^2 \varphi_{yy} \varphi_{xx} + \\ & \quad \frac{1}{2} \varphi_y^2 \varphi_{xx} \varphi_{xy}^2 - \frac{1}{2} \varphi_y^2 \varphi_{xx}^2 \varphi_{yy} - \frac{1}{2} \varphi_x^2 \varphi_{yy}^2 \varphi_{xx} + \\ & \quad \frac{1}{2} \varphi_x^2 \varphi_{yy} \varphi_{xy}^2 + \varphi_x \varphi_{xy} \varphi_y \varphi_{yy} \varphi_{xx}. \end{aligned}$$

通过求解式(18), 得到具有最小幅度的实根 t , 并插入式(17), 以获得 $\Delta p(\Delta x, \Delta y)$.

图4给出了一阶近似隐函数投影、二阶近似隐函数投影和 Distmesh 算法中的节点移动轨迹. 图中曲线 f_2 表示节点所要移动到的几何边界; 曲线 f_1 代表力平衡后移动到边界外节点所在的近似隐

函数; 虚线段 $p1 \rightarrow p4$ 表示 Distmesh 网格划分算法的节点移动轨迹; 实线段 $p1 \rightarrow p2$ 表示一阶近似隐函数投影的节点移动轨迹; 点划线段 $p1 \rightarrow p3$ 表示二阶近似隐函数投影的节点移动轨迹. 从图中可以看出 Distmesh 算法中的节点虽然投影回边界, 但较最近的边界点 $p3$ 距离偏差较大; 一阶隐函数投影方法虽在梯度方向上移动, 但节点没有投影回边界 ($p2$ 点不在曲线 f_2 上); 二阶方法则较精确地向边界方向上移动. 由此可见, 本文采用二阶近似隐函数投影方法实现节点移动.

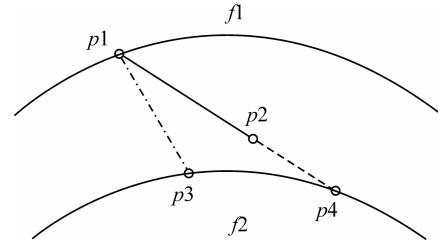


图4 3种投影方式

3 实例分析

基于上述各向异性网格算法, 分别对圆形、矩形和复杂图形进行各向异性网格划分, 并与文献[7]中的算法进行比较. 实例1为圆形的各向异性网格划分, 设定节点的期望尺寸为:

$$\begin{cases} h_1 = \min(h_{\min} + g(1 - \sqrt{x^2 + y^2}), h_{\max}), \\ h_2 = h_{\max}, \end{cases} \quad (19)$$

式中: $h_{\min} = 0.01$, $h_{\max} = 0.2$, $g = 0.3$, 旋转矩阵为 $R = (x^2 + y^2)^{-(1/2)}[x \ -y; y \ x]$.

实例2为矩形的各向异性网格划分, 设定节点的期望尺寸为:

$$\begin{cases} h_1 = h_{\max}, \\ h_2 = \min(h_{\min} + g(1 - |y|), h_{\max}), \end{cases} \quad (20)$$

式中: $h_{\min} = 0.01$, $h_{\max} = 0.2$, $g = 0.3$, 旋转矩阵为 $R = [1 \ 0; 0 \ 1]$.

实例3为较复杂图形的各向异性网格划分, 针对该图形设定节点的期望尺寸为:

$$\begin{cases} h_1 = \min(h_{\min} + g(\sqrt{x^2 + y^2} - 1), h_{\max}), \\ h_2 = h_{\max}, \end{cases} \quad (21)$$

式中: $h_{\min} = 0.01$, $h_{\max} = 0.2$, $g = 0.3$, 旋转矩阵 $R = (x^2 + y^2)^{-(1/2)}[x \ -y; y \ x]$.

对圆形的划分结果如图5所示, 从中可以看出

表 1 本文算法和文献[7]的算法结果对比

实例	对比项	节点数/个	网格单元数/个	网格单元质量分布/%					$q_{avg}/\%$
				[0,0.2)	[0.2,0.4)	[0.4,0.6)	[0.6,0.8)	[0.8,1]	
实例 1	本文	358	663	0	0.30	1.96	5.65	92.09	89.90
	文献[7]	413	774	0	1.29	4.49	4.82	89.40	87.82
实例 2	本文	318	569	0	0	0	1.01	98.99	97.56
	文献[7]	316	567	0	0	0	2.65	97.35	94.50
实例 3	本文	765	1 416	0	0	0.78	4.10	95.12	92.50
	文献[7]	799	1 478	0	0	0.68	7.10	92.22	89.01

2 种算法均实现了对圆形径向方向的精细划分. 对矩形的网格划分结果如图 6 所示, 从中可以看出 2 种算法均可实现对给定区域中网格疏密度的控制, 譬如上下边界层网格的细分. 对复杂图形的网格划分结果如图 7 所示, 从中可以看出 2 种算法可实现对多连通复杂图形的各向异性网格划分.

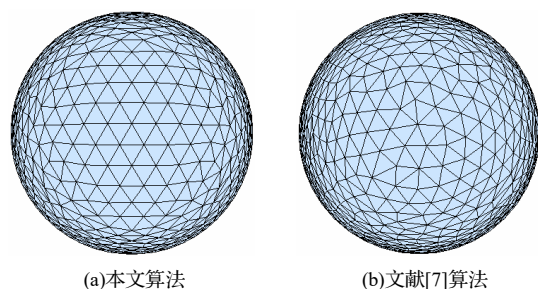


图 5 圆形的网格划分

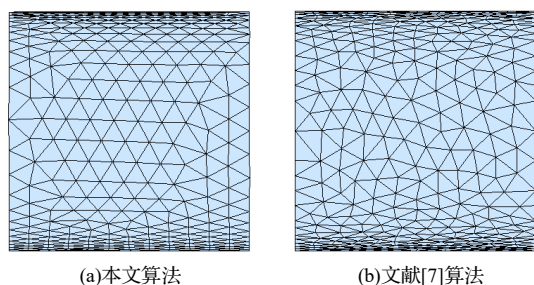


图 6 矩形的网格划分

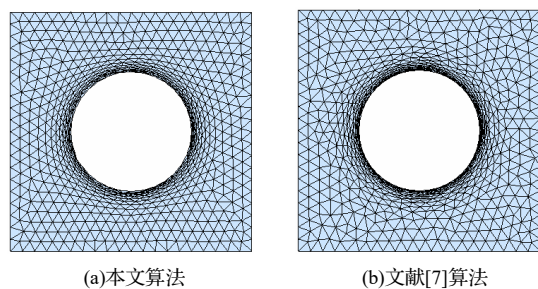


图 7 复杂图形的网格划分

表 1 为本文算法与文献[7]中算法的网格单元

质量分布和平均网格质量 q_{avg} 值(均采用式(4)计算), 从表中可看到曲率对网格平均质量的影响. 如实例 1 所示, 网格平均质量较差; 在实例 2 中, 由于矩形受曲率因素小, 故网格平均质量较优; 在实例 3 中, 虽然图形受到曲率因素影响, 但总体而言, 图形大部分节点的期望尺寸 $h_2 = h_1$, 故平均网格质量居中. 与文献[7]的算法相较, 由于本文的算法中背景网格生成时未随机剔除节点, 网格单元整体密集, 能更好地反映各向异性特征, 且采用二阶近似投影使边界节点与所求边界偏差小, 所以总体网格平均质量较好, 单元质量更趋近于 1, 但网格生成效率若有降低.

4 结语

基于黎曼度量张量, 结合各向异性 Delaunay 准则, 基于力平衡的节点优化函数和近似隐函数投影实现边界的约束处理, 减小了最后生成网格与所求几何图形的偏差值. 最终实现了各向异性网格生成算法, 在 MATLAB 环境中生成了高质量的各向异性三角网格.

参考文献:

- [1] 程中娜. 基于 ABAQUS 的智能有限元分析前处理技术研究[D]. 济南: 山东大学, 2017.
- [2] Simpson R B. Anisotropic mesh transformations and optimal error control[J]. Applied Numerical Mathematics, 1994, 14(1/2/3):183-198.
- [3] Bossen F J, Heckbert P S. A pliant method for anisotropic mesh generation[EB/OL]. [2019-03-16]. <https://people.eecs.berkeley.edu/~jrs/meshpapers/BossenHeckbert.pdf>.
- [4] Du Q, Wang D S. Anisotropic centroidal Voronoi tessellations and their applications[J]. SIAM Journal on Scientific Computing, 2005, 26(3):737-761.

- [5] 关振群, 单菊林, 顾元宪. 基于黎曼度量的复杂参数曲面有限元网格生成方法[J]. 计算机学报, 2006, 29(10): 1823-1833.
- [6] Borouchaki H, George P L, Hecht F, et al. Delaunay mesh generation governed by metric specifications. Part I. Algorithms[J]. Finite Elements in Analysis and Design, 1997, 25(1/2):61-83.
- [7] 粟一凡, 黄云清. 变度量各向异性网格生成算法和各向异性椭圆偏微分方程匹配网格[J]. 中国科学: 数学, 2016, 46(7):1037-1052.
- [8] Rippa S. Long and thin triangles can be good for linear interpolation[J]. SIAM Journal on Numerical Analysis, 1992, 29(1):257-270.
- [9] George P L, Borouchaki H. Delaunay Triangulation and Meshing[M]. Paris: Springer Press, 1998.
- [10] Huang Y Q, Su Y F, Wei H Y, et al. Anisotropic mesh generation methods based on ACVT and natural metric for anisotropic elliptic equation[J]. Science China Mathematics, 2013, 56(12):2615-2630.
- [11] Borouchaki H, George P L, Mohammadi B. Delaunay mesh generation governed by metric specifications. Part II. Applications[J]. Finite Elements in Analysis and Design, 1997, 25(1/2):85-109.
- [12] Bern M, Plassmann P. Mesh Generation[M]. New York: Elsevier Press, 2000.
- [13] Field D A. Laplacian smoothing and Delaunay triangulations[J]. Communications in Applied Numerical Methods, 1988, 4(6):709-712.
- [14] Du Q, Faber V, Gunzburger M. Centroidal Voronoi tessellations: Applications and algorithms[J]. SIAM Review, 1999, 41(4):637-676.
- [15] Wang B, Khoo B C, Xie Z Q, et al. Fast centroidal Voronoi Delaunay triangulation for unstructured mesh generation[J]. Journal of Computational and Applied Mathematics, 2015, 280:158-173.
- [16] Knupp P. Introducing the target-matrix paradigm for mesh optimization via node-movement[J]. Engineering with Computers, 2012, 28(4):419-429.
- [17] Persson P O, Strang G. A simple mesh generator in MATLAB[J]. SIAM Review, 2004, 46(2):329-345.

The anisotropic mesh generation algorithm based on Riemann metric tensor

YAN Keke, CHEN Xing*

(Faculty of Mechanical Engineering & Mechanics, Ningbo University, Ningbo 315211, China)

Abstract: In real world, some physical problems' solution has a direction-oriented feature, which exhibits solution changed significantly in a direction. An anisotropic meshing with fewer grids but better element quality is an effective way in pre-processing of the finite element method (FEM). Firstly, a background mesh under Riemann metric tensor is generated to achieve a high-quality anisotropic mesh. Secondly, the anisotropic Delaunay criterion for edge exchanges is employed. Thirdly, the nodes smoothing function is set based on force balance principle. In addition, operations related to node insertion and deletions are conducted in terms of the standardized area and length under the boundary constraints of approximate nodal projection. Finally, three examples are presented to demonstrate the feasibility of the proposed anisotropic meshing algorithm.

Key words: MATLAB; mesh generation; anisotropy; Delaunay

(责任编辑 章践立)