Discrete Geometries of Mathematics and Physics

创新实验

实验指导书

Transfinite网格生成技术实现

Yi Zhang (张仪)

: www.mathischeap.com

@: zhangyi_aero@hotmail.com

: https://github.com/mathischeap

1 实验基本信息

1.1 实验名称

Transfinite网格生成技术实现

1.2 实验指导老师联系方式

张仪(www.mathischeap.com), 数学与计算科学学院，花江慧谷4#-408A

- 手机： 199 7417 1867
- email: zhangyi_aero@hotmail.com
- QQ: 360 788 903

1.3 所属课程名称

创新实验

2 实验的目的、意义、要求

2.1 实验目的

学习理解网格生成技术中的经典方法transfinite mapping/interpolation并完成编程实现。

2.2 实验意义

- 了解网格生成的用途和意义。
- 理解transfinite interpolation/mapping的数学原理。
- 锻炼面向实际问题的编程能力。

2.3 实验要求

- 你需要具备基础的编程能力。我们推荐使用Python进行编程。如果你使用其他程序编程，你需要将我们提供的样板程序翻译成你使用的编程语言。
- 你需要完成实验报告并发回你的程序。实验过程中你可以查询网络（或询问AI等）搜集信息，但不允许直接从网络复制或要求AI生成代码或实验报告。

3 实验主页

你可以访问`https://mathischeap.com/contents/teaching/innoexp/transfinite_mesh_generation/main#course-InnoExp-transfinite_mesh_generation`查看有关本实验的所有信息。

4 实验详情

4.1 实验背景

网格是计算科学中十分重要的工具。它将计算域划分多个小的区域，称作网格单元。图1、图2和图3展示了一些实际工程中可能用到的三维网格。

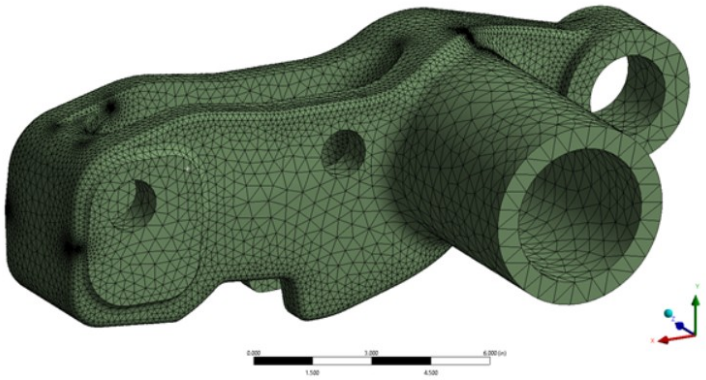

图 1: 用于计算某个机械部件内部应力分布的四面体网格。

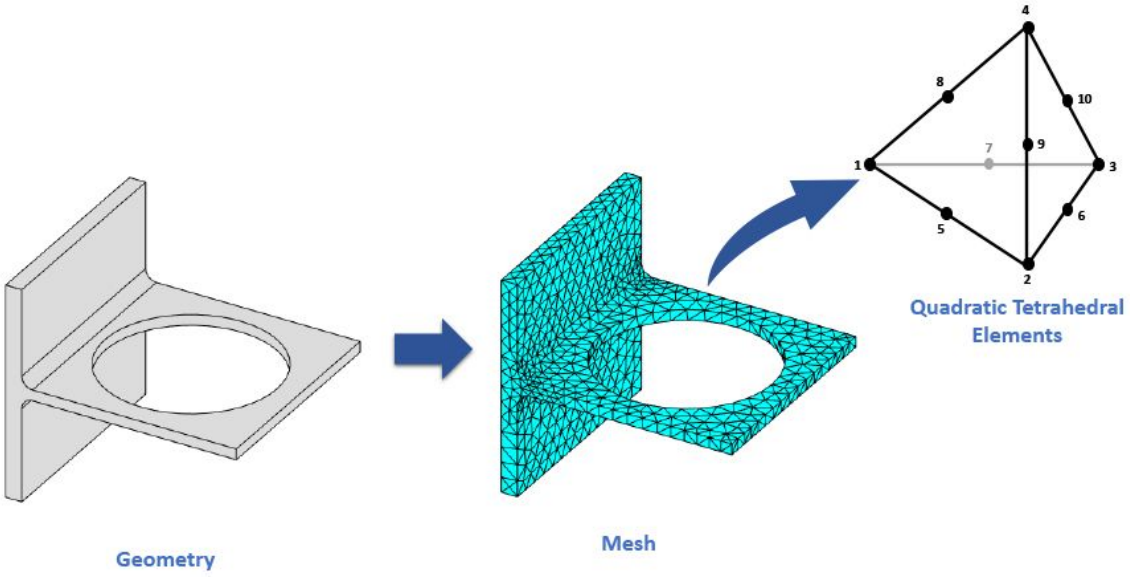

图 2: 计算部件应力分布的四面体网格（中间图）。

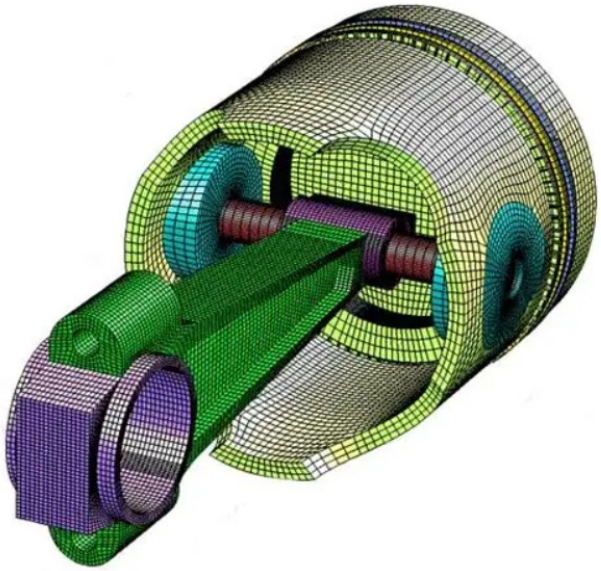

图 3: 用于发动机活塞仿真的六面体网格。

网格生成技术多种多样[1]。其中有一类方法被称作映射法网格生成技术。它将一个规则参考域内的已知网格映射到真实计算域内从而在生成所需的网格。这一类方法里，最重要的也是最简单的方法之一就是transfinite interpolation/mapping(以下统一称作transfinite mapping)网格生成方法[2, 3]。此方法诞生于上世纪七八十年代，在针对较为规则的简单计算域时生成效率高，网格质量好，时至今日依然是非常常用的网格生成方法。本实验将介绍此方法的二维版本，并将其用于生成一些简单二维计算域内的网格。三维情况更常见、更复杂但数学原理相似，感兴趣的同学可以查阅相关资料自学。

4.2 Transfinite mapping

在参考空间（或称作逻辑空间，logical space）下，我们建立一个参考坐标系(ξ,η)。参考域被定义成参考坐标系下的单位立方体$(\xi,\eta):=[0,1]^2$。它的四个角点和四条边的命名和分布如图4中左侧的示意图。我们将实际计算发生的空间称为物理空间。在物理空间下，我们建立一个物理坐标系(x,y)。实际计算发生的区域就被称为物理（计算）域。Transfinite mapping就是将参考域映射至物理域的一个映射，如图4所示。

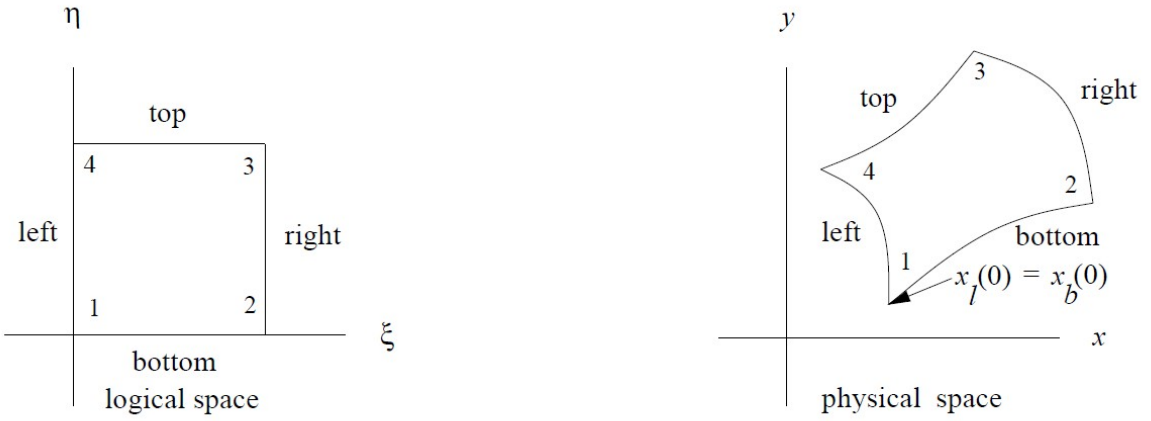

图 4: 左：参考空间中参考坐标系(ξ,η)下的参考域$[0,1]^2$。右：物理空间中物理坐标系(x,y)下的物理（计算）域的示意图。Transfinite mapping将参考域映射至物理域，且（必须）保持拓扑关系不发生变化。如，逆时针方向，参考域和物理域中角点和边的顺序都为：$1\to$ bottom $\to 2\to$ right $\to 3\to$ top $\to 4\to$ left $\to 1$。本图引用自[1]。

为了构造transfinite mapping，我们需要首先构造参考域的4条边到物理域的4条边的映射，也就是，

$$\text{底边至底边(bottom):}\quad \begin{bmatrix}x\\y\end{bmatrix}=\Phi_{\mathrm{b}}(\xi)=\begin{bmatrix}\Phi_{\mathrm{b}}^{x}(\xi)\\\Phi_{\mathrm{b}}^{y}(\xi)\end{bmatrix},\qquad \xi\in[0,1],$$

$$\text{顶边至顶边(top):}\quad \begin{bmatrix}x\\y\end{bmatrix}=\Phi_{\mathrm{t}}(\xi)=\begin{bmatrix}\Phi_{\mathrm{t}}^{x}(\xi)\\\Phi_{\mathrm{t}}^{y}(\xi)\end{bmatrix},\qquad \xi\in[0,1],$$

$$\text{左边至左边(left):}\quad \begin{bmatrix}x\\y\end{bmatrix}=\Phi_{\mathrm{l}}(\eta)=\begin{bmatrix}\Phi_{\mathrm{l}}^{x}(\eta)\\\Phi_{\mathrm{l}}^{y}(\eta)\end{bmatrix},\qquad \eta\in[0,1],$$

$$\text{右边至右边(right):}\quad \begin{bmatrix}x\\y\end{bmatrix}=\Phi_{\mathrm{r}}(\eta)=\begin{bmatrix}\Phi_{\mathrm{r}}^{x}(\eta)\\\Phi_{\mathrm{r}}^{y}(\eta)\end{bmatrix},\qquad \eta\in[0,1].$$

当给出这四个边的映射时，我们可以通过角点的映射验证它们是否合理。比如，角点1，它可由底边映射在$\xi=0$时得到，也可以由左边映射在$\eta=0$时得到，即

$$\begin{bmatrix}x_1\\y_1\end{bmatrix}=\begin{bmatrix}\Phi_{\mathrm{b}}^{x}(0)\\\Phi_{\mathrm{b}}^{y}(0)\end{bmatrix}\quad\text{且}\quad\begin{bmatrix}x_1\\y_1\end{bmatrix}=\begin{bmatrix}\Phi_{\mathrm{l}}^{x}(0)\\\Phi_{\mathrm{l}}^{y}(0)\end{bmatrix}.$$

那么我们可以分别计算这两个映射，并检查它们是否得到相同的角点1的坐标(x_1,y_1)。同理，我们可以使用其他角点来验证。当所有角点的验证都通过时，我们就能（基本上）确定上述4个边的映

射是满足transfinite mapping的基本要求的。接下来，我们可以得到如下的transfinite mapping，

$$\begin{aligned}\boldsymbol{x} = &(1-\eta)\Phi_{\mathrm{b}}(\xi) + \eta\Phi_{\mathrm{t}}(\xi) + (1-\xi)\Phi_{\mathrm{l}}(\eta) + \xi\Phi_{\mathrm{r}}(\eta)\\ &- \xi\eta\Phi_{\mathrm{t}}(1) - \xi(1-\eta)\Phi_{\mathrm{b}}(1) - \eta(1-\xi)\Phi_{\mathrm{t}}(0) - (1-\xi)(1-\eta)\Phi_{\mathrm{b}}(0).\end{aligned} \tag{1}$$

注意，上式中，我们把两个分量写在了一起，并且，显然$\Phi_{\mathrm{t}}(1)$, $\Phi_{\mathrm{b}}(1)$, $\Phi_{\mathrm{t}}(0)$, $\Phi_{\mathrm{b}}(0)$分别代表物理域的角点3，2，4，1的坐标。如果我们把(1)写成分量形式，我们有

$$\begin{aligned}x = \Phi^x(\xi,\eta) = &(1-\eta)\Phi^x_{\mathrm{b}}(\xi) + \eta\Phi^x_{\mathrm{t}}(\xi) + (1-\xi)\Phi^x_{\mathrm{l}}(\eta) + \xi\Phi^x_{\mathrm{r}}(\eta)\\ &- \xi\eta x_3 - \xi(1-\eta)x_2 - \eta(1-\xi)x_4 - (1-\xi)(1-\eta)x_1,\end{aligned} \tag{2}$$

$$\begin{aligned}y = \Phi^y(\xi,\eta) = &(1-\eta)\Phi^y_{\mathrm{b}}(\xi) + \eta\Phi^y_{\mathrm{t}}(\xi) + (1-\xi)\Phi^y_{\mathrm{l}}(\eta) + \xi\Phi^y_{\mathrm{r}}(\eta)\\ &- \xi\eta y_3 - \xi(1-\eta)y_2 - \eta(1-\xi)y_4 - (1-\xi)(1-\eta)y_1,\end{aligned} \tag{3}$$

其中(x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)分别为物理域的角点1，2，3，4的坐标。(2)和(3)一起就定义了我们寻找的transfinite mapping Φ,

$$\begin{bmatrix}x\\ y\end{bmatrix} = \Phi(\xi,\eta) = \begin{bmatrix}\Phi^x(\xi,\eta)\\ \Phi^y(\xi,\eta)\end{bmatrix}.$$

任务1：编程实现上述二维transfinite mapping

> 你需要编写一个程序计算二维transfinite mapping。程序的输入变量包括4个边的映射函数（或者它们的总计8个分量的映射函数），以及在参考域下的坐标(ξ,η)。输出变量为(ξ,η)在物理域下的映射坐标(x,y)。注意，你可以自己设计程序的形式。无论是采用面向过程还是面向对象的编程方法，我们都应该能够设计出合理的程序结构实现transfinite mapping。在完成编程后，你可以自己设计一些输入变量用于初步验证你的程序是否正确。或者，你可以跳过验证、继续下一节中的实验内容。它将教你如何系统性验证你编写的transfinite mapping是否正确。

4.3 网格生成和可视化

下面，我们演示如果使用transfinite mapping来生成网格。这个过程其实非常简单，我们只需要把参考域中的网格使用构造好的transfinite mapping映射至物理域中即可。在本实验的主页上，你可以下载到以下Python程序文件,

`transfinite_mapping_helper.py`.

它包含了两个函数：`reference_mesh_generator`和`visualize_physical_mesh`。

给定一个不小于2的正整数N（推荐取$N>5$以获得足够密的网格），运行

`reference_mesh_generator`(N)

将给出两个二维数组ξ和η，它们就是参考域中的网格数据。将它们作为输入变量传递给你的transfinite mapping程序，即可得到物理域下的网格数据x和y。继续将x和y作为输入变量传递给网格可视化函数，

`visualize_physical_mesh`(x,y)

就可以完成网格的可视化。总结而言，整个程序流为

$$\xi, \eta = \texttt{reference\_mesh\_generator}(N)$$
$$\downarrow$$
$$x, y = \texttt{transfinite\_mapping}(\xi, \eta)$$
$$\downarrow$$
$$\texttt{visualize\_physical\_mesh}(x, y)$$

其中`transfinite_mapping`(ξ, η)就是你自己编写的部分。

任务2：网格生成与可视化

使用如下计算域，结合上述网格生成与可视化方法，验证你的transfinite mapping程序的正确性。

- 计算域一：此计算域为一个矩形

$$[0, 2] \times [0, 1].$$

  可以看到，这个计算域非常简单。你需要自己找出它的四条边所对应的映射函数。

- 计算域二：此计算域的除去顶边($y = 1$)以外，另外三条边与计算域一相同。计算域二的顶边映射函数为：

$$\text{顶边至顶边(top):} \quad \begin{bmatrix} x \\ y \end{bmatrix} = \Phi_{\mathrm{t}}(\xi) = \begin{bmatrix} 2\xi \\ 1 + 0.25\sin(2\pi\xi) \end{bmatrix}, \quad \xi \in [0, 1].$$

- 计算域三：此计算域除去底边($y = 0$)以外，另外三条边与计算域二相同。计算域三的底边映射函数为：

$$\text{底边至底边(bottom):} \quad \begin{bmatrix} x \\ y \end{bmatrix} = \Phi_{\mathrm{b}}(\xi) = \begin{bmatrix} 2\xi^2 \\ -0.1\xi(\xi - 1) \end{bmatrix}, \quad \xi \in [0, 1].$$

5 总结

在完成上述实验任务后，你需要完成一份完整的实验报告并附上你编写的代码。

References

[1] P. Knupp, S. Steinberg, Fundamentals of Grid Generation, CRC Press, 1993.

[2] W. J. Gordon, L. C. Thiel, Transfinite mappings and their application to grid generation, Applied Mathematics and Computation 10-11 (1982) 171–233. doi:https://doi.org/10.1016/0096-3003(82)90191-6.

[3] W. J. Gordon, C. A. Hall, Construction of curvilinear co-ordinate systems and applications to mesh generation, International Journal for Numerical Methods in Engineering 7 (4) (1973) 461–477. doi:https://doi.org/10.1002/nme.1620070405.