



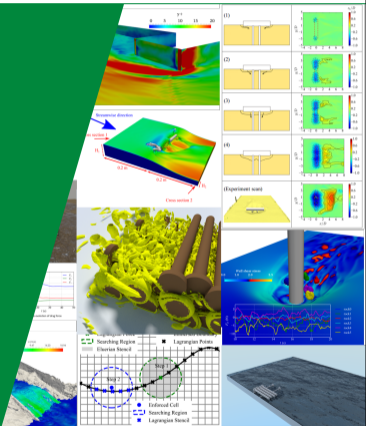
中国农业大学 流体机械与流体工程系

2024年春季《计算流体力学编程实践》

# 第三章 网格处理 blockMesh

徐云成

✉ ycxu@cau.edu.cn



2024年3月28日

- ▶ 计算网格分类和要求
- ▶ OpenFOAM® 中的网格工具
- ▶ 如何使用blockMesh 划分网格



- ▶ 一个计算网格是指用来描述模拟中的空间域，包括边界离散和空间离散
- ▶ 生成网格并不简单
- ▶ 生成网格通常占用整个CFD模拟一半以上的精力



## 网格类型:

- ▶ 结构化网格 structured grid / mesh
- ▶ 非结构化网格 unstructured mesh / grid
- ▶ 重叠网格 Overset meshes

grid和mesh的区别在于，grid通常指规则的、结构化的网格布局，而mesh则更灵活，可以包括规则和非规则的网格，适应各种复杂的几何形状。

简言之，grid重在规则性，mesh重在适应性。从另一个角度：

- ▶ 贴体网格 body-fitted mesh
- ▶ 浸没边界法 immersed boundary method on a fixed background mesh
- ▶ 无网格法 mesh-less methods, e.g., LBM, SPH



网格类型:

- ▶ 结构化网格 structured grid / mesh
- ▶ 非结构化网格 unstructured mesh / grid
- ▶ 重叠网格 Overset meshes

grid和mesh的区别在于, grid通常指规则的、结构化的网格布局, 而mesh则更灵活, 可以包括规则和非规则的网格, 适应各种复杂的几何形状。

简言之, grid重在规则性, mesh重在适应性。从另一个角度:

- ▶ 贴体网格 body-fitted mesh
- ▶ 浸没边界法 immersed boundary method on a fixed background mesh
- ▶ 无网格法 mesh-less methods, e.g., LBM, SPH



网格类型:

- ▶ 结构化网格 structured grid / mesh
- ▶ 非结构化网格 unstructured mesh / grid
- ▶ 重叠网格 Overset meshes

grid和mesh的区别在于, grid通常指规则的、结构化的网格布局, 而mesh则更灵活, 可以包括规则和非规则的网格, 适应各种复杂的几何形状。

简言之, grid重在规则性, mesh重在适应性。从另一个角度:

- ▶ 贴体网格 body-fitted mesh
- ▶ 浸没边界法 immersed boundary method on a fixed background mesh
- ▶ 无网格法 mesh-less methods, e.g., LBM, SPH



如何定义网格尺度和质量？

- ▶ 最重要的是你希望通过模拟捕捉的物理现象！
- ▶ 哪些区域需要考虑加密网格？
  - 物理问题最关心的区域
  - 物理量变化较快（梯度较大）的区域
- ▶ 维度：二维还是三维？
- ▶ 对于湍流模型，你想要如何模拟？你能调用多少计算量？
  - DNS, LES or RANS
  - 如何处理大梯度（sharp gradient）区域，比如无滑移壁面？



- ▶ 准备几何造型
  - CAD
  - 三维扫描+数字化处理
- ▶ 几何造型的加工处理
  - 加密处理 (Refinement) 通过插值等方法
  - 简化处理 (Coarsening) 去除一些不必要的细节
- ▶ 在网格生成软件中导入几何表面网格 (surface mesh)
  - 表面网格通常会用来定义需要模拟的边界, 会分为不同边界 (patches), 比如进口、出口等
  - 一般而言, 如果几何造型比较简单, 可以在网格生成软件中直接生成表面网格
- ▶ 生成体网格
  - 计算域内的网格单元不会发生重叠
  - 网格位置决定了离散解所在位置, 因此在网格划分前, 需要对求解域有一些认识 (预判)





- ▶ 网格纵横比 Cell aspect ratio
- ▶ 非正交性 Non-orthogonality
- ▶ 畸变率/偏斜率/偏度 Skewness



```
Foam::scalar Foam::primitiveMeshTools::faceSkewness
(
    const primitiveMesh& mesh,
    const pointField& p,
    const vectorField& fCtrs,
    const vectorField& fAreas,

    const label facei,
    const point& ownCc,
    const point& neiCc
)
{
    vector Cpf = fCtrs[facei] - ownCc;
    vector d = neiCc - ownCc;

    // Skewness vector
    vector sv =
        Cpf
        - ((fAreas[facei] & Cpf)/((fAreas[facei] & d) + rootVSmall))*d;
    vector svHat = sv/(mag(sv) + rootVSmall);

    // Normalisation distance calculated as the approximate distance
    // from the face centre to the edge of the face in the direction
    // of the skewness
    scalar fd = 0.2*mag(d) + rootVSmall;
    const face& f = mesh.faces()[facei];
    forAll(f, pi)
    {
        fd = max(fd, mag(svHat & (p[f[pi]] - fCtrs[facei])));
    }

    // Normalised skewness
    return mag(sv)/fd;
}
```

mesh:网格

p:网格顶点

fCtrs: 面 (face) 中心点

fAreas:面的面积向量

facei:面序号

ownCc:该面相邻的一个网格中心点

neiCc:该面相邻的另一个网格中心点

mag():magnitude

&:点乘

rootVSmall:非常小的值 $10^{-38}$ ??

face:面顶点序号



- ▶ 静网格 Static
- ▶ 动网格, 网格变形(mesh deformation)或拓扑结构发生变化(topological change)
  - 几何结构在计算域内移动
  - 适应性网格加密/稀疏化(Adaptive mesh refinement/coarsening)
    - alpha、梯度、模型误差
- ▶ 常见的拓扑结构变化
  - 接触边界、脱离边界 (Attach/detach boundary)
  - 增加、移除网格层 (Cell layer addition/removal)
  - 滑移界面(Sliding interface)



- ▶ 源代码在 `applications/utilities/mesh`
- ▶ 网格划分 `applications/utilities/mesh/generation`:
  - `blockMesh`
  - `snappyHexMesh`
  - `foamyMesh`
- ▶ 网格转换 (`applications/utilities/mesh/conversion`):
  - `fluent3DmeshToFoam`
  - `fluentMeshToFoam`
  - `star4ToFoam`
  - `gambitToFoam`
  - `cfx4ToFoam`
  - `ansysToFoam`



- ▶ 源代码在 `applications/utilities/mesh`
- ▶ 网格划分 `applications/utilities/mesh/generation`:
  - `blockMesh`
  - `snappyHexMesh`
  - `foamyMesh`
- ▶ 网格转换 (`applications/utilities/mesh/conversion`):
  - `fluent3DmeshToFoam`
  - `fluentMeshToFoam`
  - `star4ToFoam`
  - `gambitToFoam`
  - `cfx4ToFoam`
  - `ansysToFoam`



- ▶ 源代码在 `applications/utilities/mesh`
- ▶ 网格划分 `applications/utilities/mesh/generation`:
  - `blockMesh`
  - `snappyHexMesh`
  - `foamyMesh`
- ▶ 网格转换 (`applications/utilities/mesh/conversion`):
  - `fluent3DMeshToFoam`
  - `fluentMeshToFoam`
  - `star4ToFoam`
  - `gambitToFoam`
  - `cfx4ToFoam`
  - `ansysToFoam`



- ▶ 其他网格工具 applications/utilities/mesh/advanced:
  - refineWallLayer 加密靠近边界的网格
  - collapseEdges 整合同一条线上的边
  - ...
- ▶ 网格处理 applications/utilities/mesh/manipulation:
  - checkMesh 检查网格质量
  - topoSet 在 cellSets/faceSets/pointSets 进行操作（创建 create、删除 delete、反选 invert、剔除 subset）
  - refineMesh 通常是根据 topoSet 得到的 cellSets 进行网格加密
  - transformPoints 平移 translate、旋转 rotate、缩放 scale 网格顶点
  - moveMesh 网格在边界上根据某种既定方式进行变形
  - createPatch 根据已选择的边界面（boundary face）创建新的计算边界（patch）



- ▶ 其他网格工具 `applications/utilities/mesh/advanced`:
  - `refineWallLayer` 加密靠近边界的网格
  - `collapseEdges` 整合同一条线上的边
  - ...
- ▶ 网格处理 `applications/utilities/mesh/manipulation`:
  - `checkMesh` 检查网格质量
  - `topoSet` 在 `cellSets/faceSets/pointSets` 进行操作（创建 `create`、删除 `delete`、反选 `invert`、剔除 `subset`）
  - `refineMesh` 通常是根据 `topoSet` 得到的 `cellSets` 进行网格加密
  - `transformPoints` 平移 `translate`、旋转 `rotate`、缩放 `scale` 网格顶点
  - `moveMesh` 网格在边界上根据某种既定方式进行变形
  - `createPatch` 根据已选择的边界面（`boundary face`）创建新的计算边界（`patch`）





- ▶ OpenFOAM® 还有很多用来处理表面几何数据 (surfaces, such as STL, OBJ) 的工具
- ▶ 源代码在 `applications/utilities/surface`:
  - `surfaceCheck` 检查表面网格的拓扑结构, 包括相邻网格的法方向
  - `surfaceConvert` 转换格式
  - `surfaceTransformPoints` translate、rotate、scale表面网格顶点
  - `surfaceSmooth` 对网格顶点使用Laplacian smoothing, 进行平滑处理
  - `surfaceCoarsen` 稀疏化处理

更多工具可见

<https://cfd.direct/openfoam/user-guide/v8-standard-utilities/#x14-1110003.6.2>



- ▶ constant/polyMesh中常见文件:

boundary faces neighbour owner points

## points

```
882
(
  (0 0 0)
  (0.005 0 0)
  (0.01 0 0)
  (0.015 0 0)
  ...
)
```

## faces

```
1640
(
  4(1 22 463 442)
  4(21 462 463 22)
  4(2 23 464 443)
  4(22 463 464 23)
  ...
)
```

## owner

```
1640
(
  0
  0
  1
  1
  ...
)
```

## neighbour

```
760
(
  1
  20
  2
  21
  ...
)
```

- ▶ 同一个face, owner序号要小于neighbour
- ▶ face法向是远离owner



- ▶ constant/polyMesh中常见文件:

boundary faces neighbour owner points

## points

```
882
(
  (0 0 0)
  (0.005 0 0)
  (0.01 0 0)
  (0.015 0 0)
  ...
)
```

## faces

```
1640
(
  4(1 22 463 442)
  4(21 462 463 22)
  4(2 23 464 443)
  4(22 463 464 23)
  ...
)
```

## owner

```
1640
(
  0
  0
  1
  1
  ...
)
```

## neighbour

```
760
(
  1
  20
  2
  21
  ...
)
```

- ▶ 同一个face, owner序号要小于neighbour
- ▶ face法向是远离owner



```
3  
(  
    movingWall  
    {  
        type                wall;  
        inGroups            List<word> 1(wall);  
        nFaces              20;  
        startFace          760;  
    }  
    fixedWalls  
    {  
        type                wall;  
        inGroups            List<word> 1(wall);  
        nFaces              60;  
        startFace          780;  
    }  
    ...
```



- ▶ blockMesh是OpenFOAM® 中最基础的网格划分工具，主要是通过system/blockMeshDict一个脚本字典文件(dictionary file)生成网格，主要是在constant/polyMesh中输出points, faces, cells, boundary。
- ▶ 基本原理：将计算域划分为一个或多个六面体块(hexahedral blocks)，体块的边可以是直线(straight line)、弧线(arc)、样条曲线(spline)，每个方向上定义网格划分密度。
- ▶ 一个体块一般可以由8个顶点组成，也可以少于8个顶点（比较少见，一般不考虑）



```
convertToMeters 1;
```

```
vertices  
(  
    ...  
);  
  
blocks  
(  
    ...  
);  
  
edges  
(  
    hex (0 1 2 3 4 5 6 7)(10 10 1)simpleGrading(1.0 1.0 1.0)  
);  
  
patches  
(  
    patch inlet ( (4 7 3 0) )  
    ...  
);
```

- ▶ convertToMeters 顶点坐标缩放因子: 0.001 scales to mm
- ▶ vertices 顶点坐标: (0 0 0)
- ▶ edges 用于arc、spline边: arc 1 4 (0.939 0.342 -0.5)
- ▶ block 顶点序号组合和网格尺寸:  
hex (0 1 2 3 4 5 6 7)(10 10 1)simpleGrading(1.0 1.0 1.0)
- ▶ patches 边界条件(patches):  
patch inlet ( (4 7 3 0) )
- ▶ mergePatchPairs 需要合并的边界条件, 暂不考虑



```
vertices
```

```
(
```

```
    ( 0    0    0 ) // vertex number 0
```

```
    ( 1    0    0.1) // vertex number 1
```

```
    ( 1.1  1    0.1) // vertex number 2
```

```
    ( 0    1    0.1) // vertex number 3
```

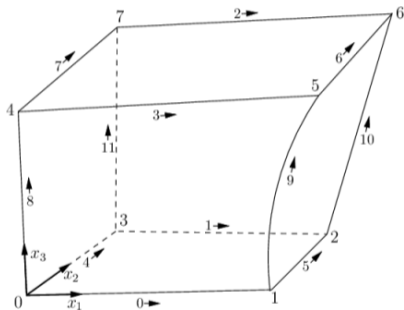
```
    (-0.1 -0.1  1 ) // vertex number 4
```

```
    ( 1.3  0    1.2) // vertex number 5
```

```
    ( 1.4  1.1  1.3) // vertex number 6
```

```
    ( 0    1    1.1) // vertex number 7
```

```
);
```



```
edges
(
    arc 1 5 (1.1 0.0 0.5) // 顶点1和顶点5之间的1个插值点
);
```

```
edges
(
    arc 1 5 25 (0 1 0) // 25 degrees, y-normal
);
```

还有spline, polyLine, BSpline, 对应多个插值点





```
blocks
(
    hex (0 1 2 3 4 5 6 7)      // vertex numbers
    (10 10 10)                 // numbers of cells in each direction
    simpleGrading (1 2 3)      // cell expansion ratios
);
```

另一种:

```
edgeGrading (1 1 1 1 2 2 2 2 3 3 3 3)
```



```
blocks
(
  hex (0 1 2 3 4 5 6 7) (100 300 100)
  simpleGrading
  (
    1          // x-direction expansion ratio
    (
      (0.2 0.3 4) // 20% y-dir, 30% cells, expansion = 4
      (0.6 0.4 1) // 60% y-dir, 40% cells, expansion = 1
      (0.2 0.3 0.25) // 20% y-dir, 30% cells, expansion = 0.25
    )
    3          // z-direction expansion ratio
  )
);
```



```
boundary // keyword
(
  inlet // patch name
  {
    type patch; // patch type for patch 0
    faces
    (
      (0 4 7 3) // block face in this patch
    );
  } // end of 0th patch definition
  walls // patch name
  {
    type wall; // patch type for patch 1
    faces
    (
      (0 1 5 4)
      (0 3 2 1)
      (3 7 6 2)
      (4 5 6 7)
    );
  }
)

boundary
(
  patch inlet
  {
    (0 4 7 3)
  }
  wall walls
  {
    (0 1 5 4)
    (0 3 2 1)
    (3 7 6 2)
    (4 5 6 7)
  }
  ...
);
```



```
boundary                // keyword
(
  inlet                  // patch name
  {
    type patch;         // patch type for patch 0
    faces
    (
      (0 4 7 3) // block face in this patch
    );
  } // end of 0th patch definition
  walls                  // patch name
  {
    type wall;          // patch type for patch 1
    faces
    (
      (0 1 5 4)
      (0 3 2 1)
      (3 7 6 2)
      (4 5 6 7)
    );
  }
}

boundary
(
  patch inlet
  {
    (0 4 7 3)
  }
  wall walls
  {
    (0 1 5 4)
    (0 3 2 1)
    (3 7 6 2)
    (4 5 6 7)
  }
  ...
);
```



```
left
{
    type                cyclic;
    neighbourPatch      right;
    faces                ((0 4 7 3));
}
right
{
    type                cyclic;
    neighbourPatch      left;
    faces                ((1 5 6 2));
}
```



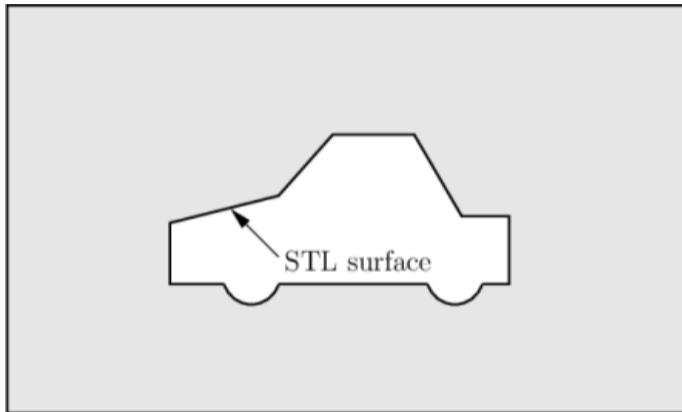


图: 几何表面与计算域

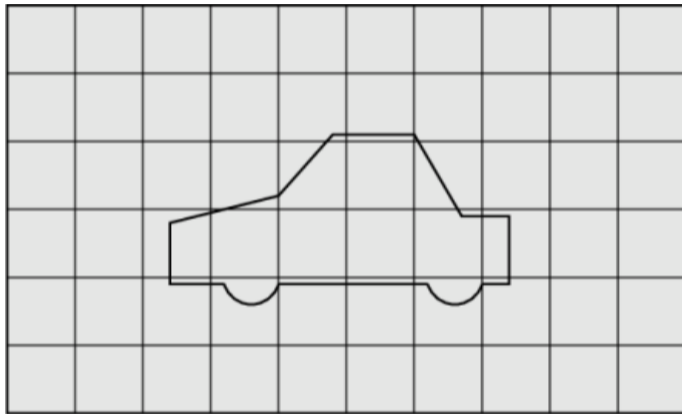


图: 生成背景网格 (一般用blockMesh)

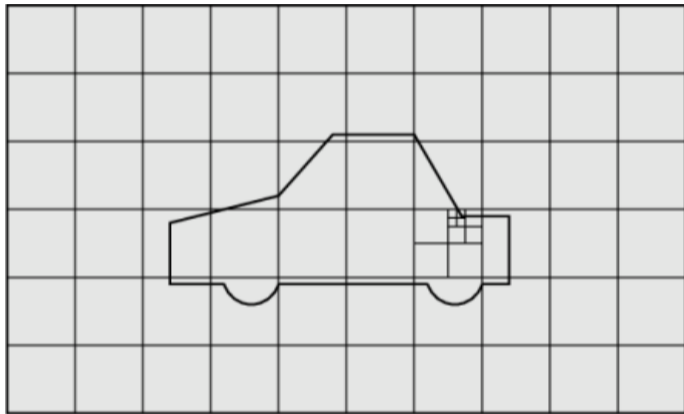


图: 对于局部细节进行加密 feature edge



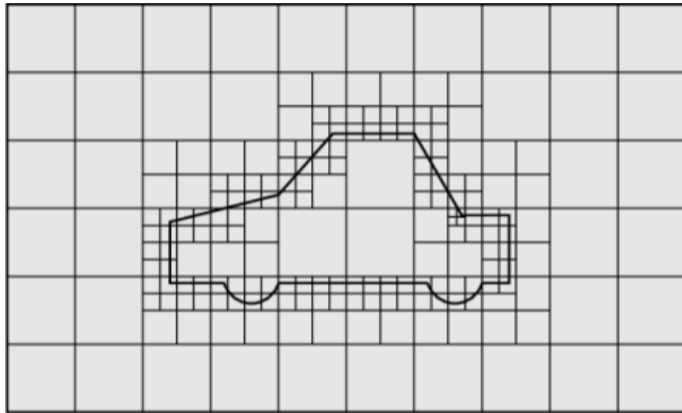


图: 识别出相交网格并进行加密

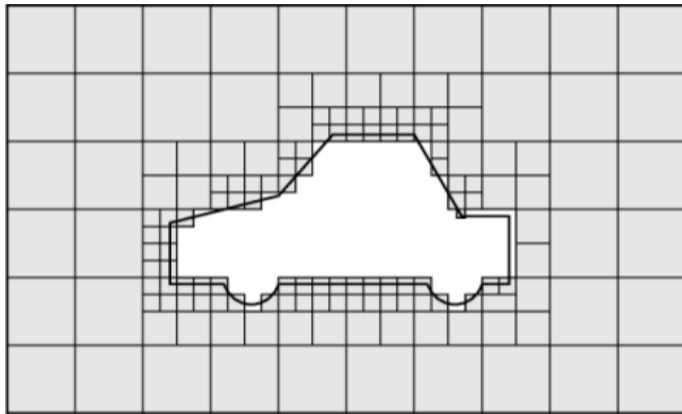


图: 移除几何内部网格

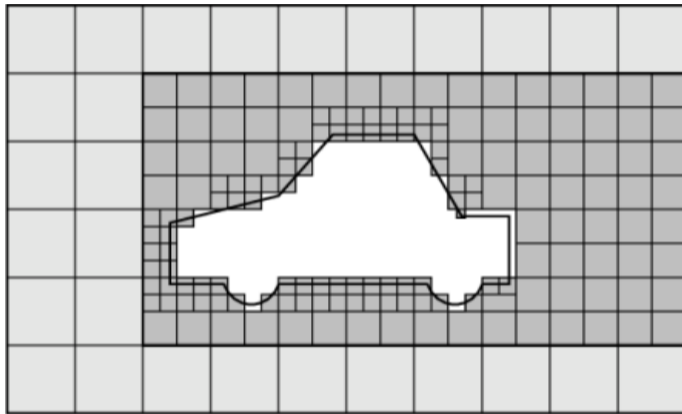


图: 对指定区域进行加密

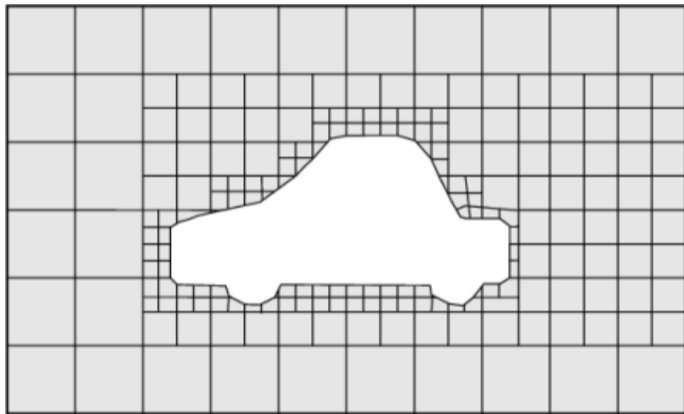


图: 移动边界网格顶点, 使之与表面几何网格重合

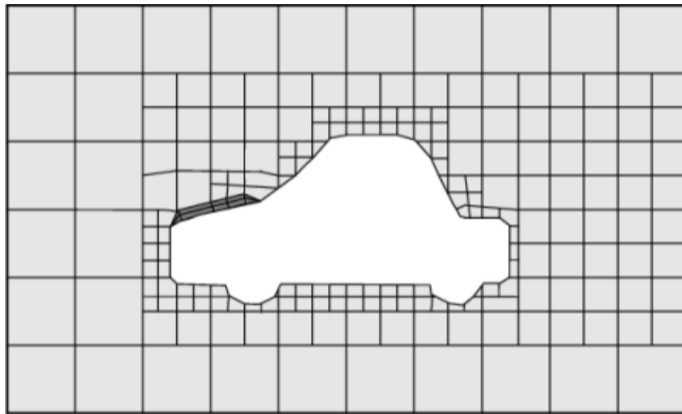


图: 生成边界层网格 (阴影部分)

- ▶ 更新gitee.com代码平台  
`git fetch --all; git pull`
- ▶ 演示3个block情况的blockMeshDict
- ▶ 解读snappyHexMesh



Thank you.

欢迎私下交流，请勿私自上传网络，谢谢！

