SHANGHAI JIAO TONG UNIVERSITY

大规模工程计算

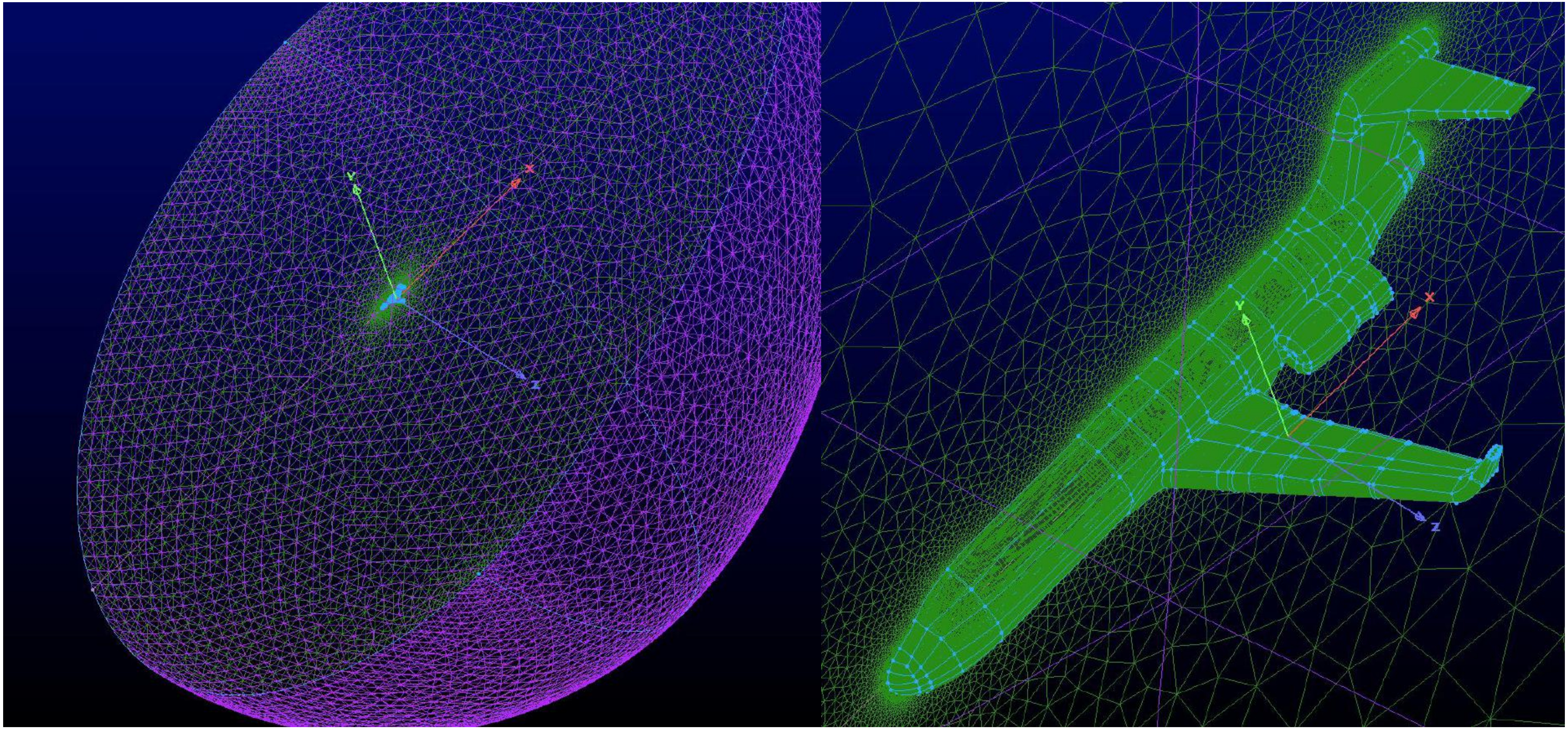
——流体力学并行计算

印子斐

长聘教轨副教授

上海交通大学航空航天学院

大规模工程计算?



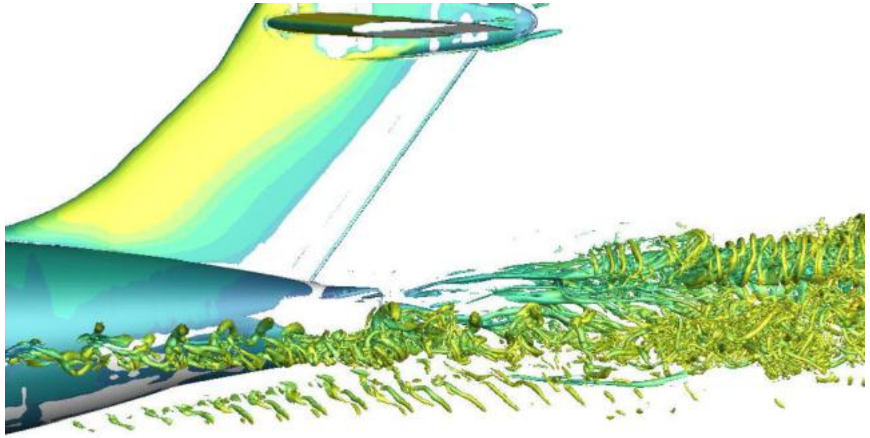
目录
CONTENTS

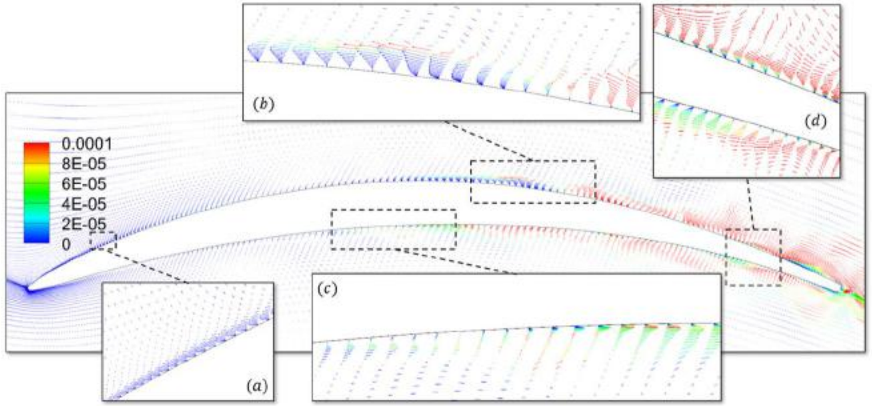
（相关内容见：AE9601 航空航天高性能计算；MS3406 粘性流体力学； AE2702 计算方法）

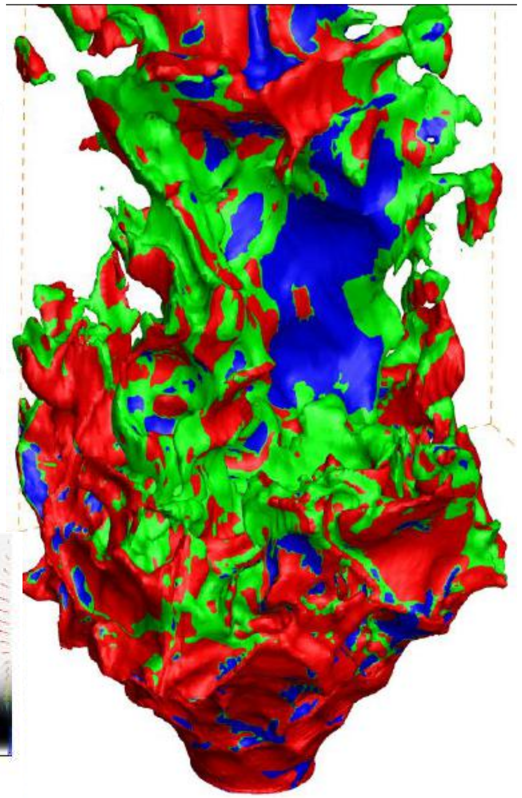
PART 01

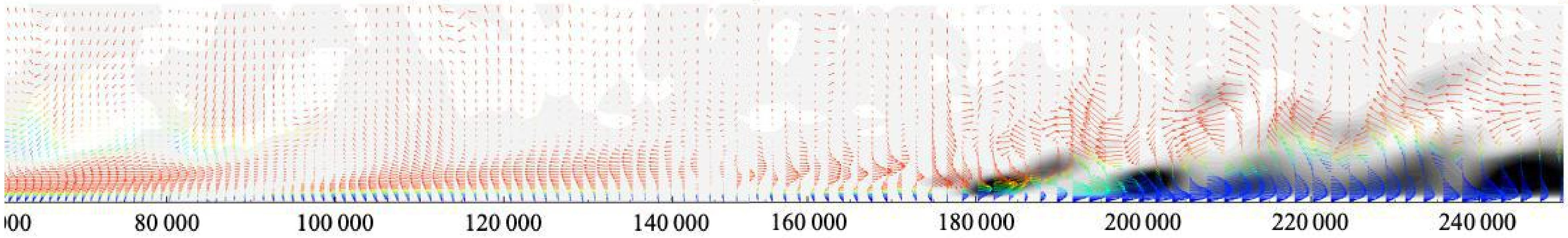
计算流体力学简介

什么是计算流体力学（CFD）







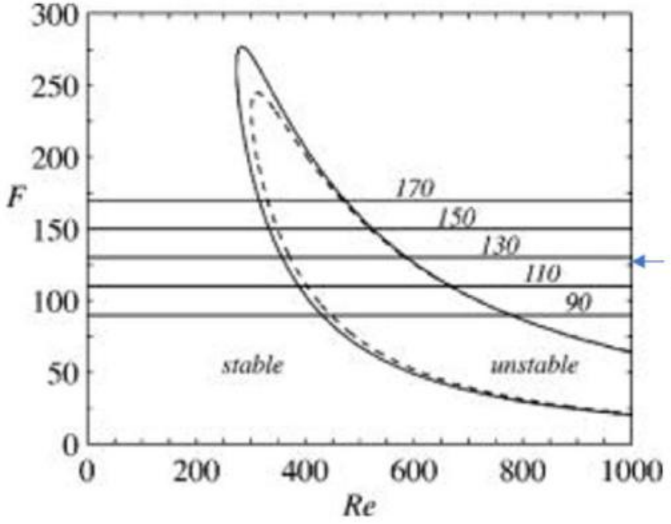


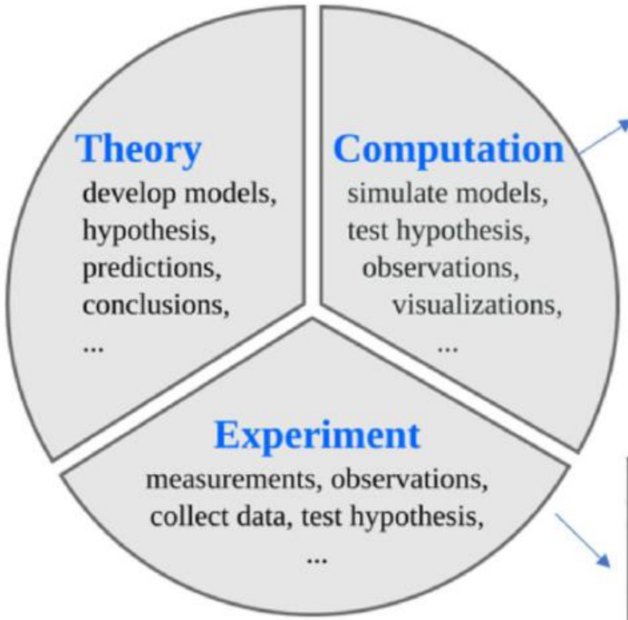
Colorful **F**luids **D**ynamics ?

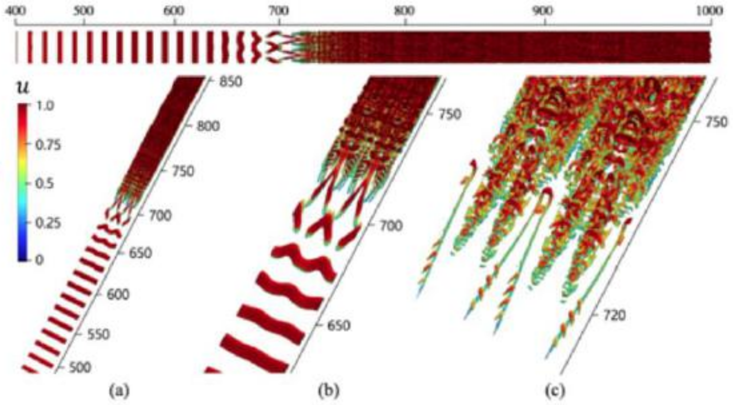
Computational **F**luids **D**ynamics

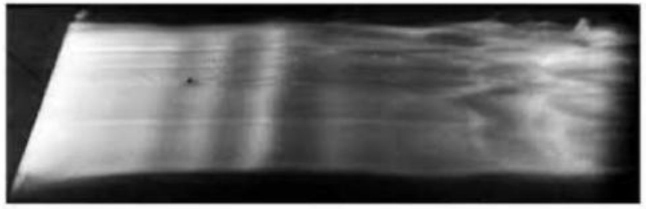
计算流体力学有什么用?

数值模拟是我们研究物理本质、解决工程问题的有力工具









- **计算流体力学的重要组成部分：**
  - 基本公式：Navier-Stokes 方程，滤波或时间平均后的Navier-Stokes方程
  - 数值离散：偏微分方程的数值解
  - 计算机科学：高性能计算

可压缩流动Navier-Stokes方程

$$
\begin{cases}
\frac{\partial}{\partial x_i}(\rho U_i) = 0 \\
\frac{\partial \rho U_i}{\partial t} + \frac{\partial}{\partial x_j}(\rho U_j U_i) = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial U_i}{\partial x_j}\right) \\
\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_j}(\rho U_j E) + \frac{\partial}{\partial x_j}(U_j p) = -\frac{\partial q_j}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\tau_{ij} \frac{\partial U_i}{\partial x_j}\right)
\end{cases}
$$

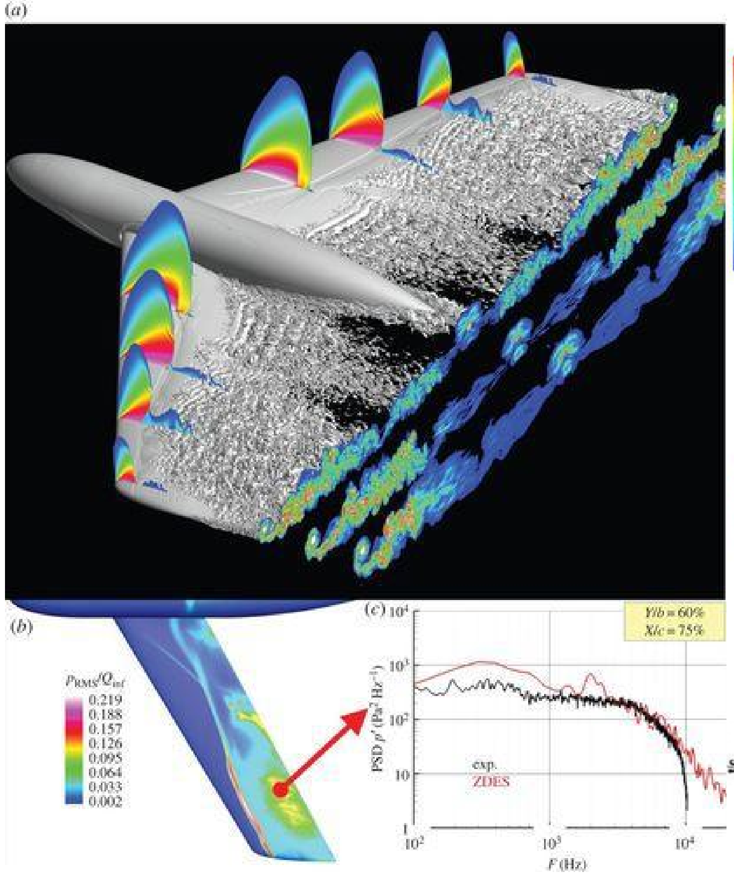
不可压缩流动的Navier-Stokes方程

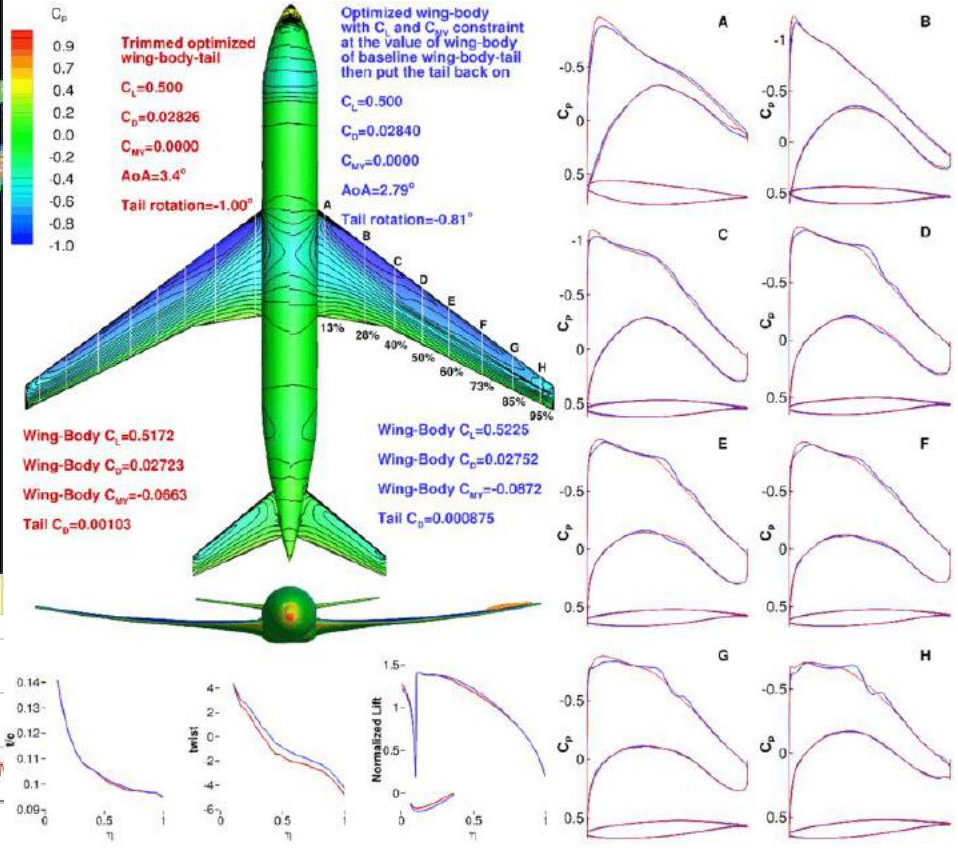
$$
\begin{cases}
\frac{\partial U_i}{\partial x_i} = 0 \\
\frac{\partial U_i}{\partial t} + \mathrm{U}_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + v \frac{\partial^2 U_i}{\partial x_j \partial x_j}
\end{cases}
$$

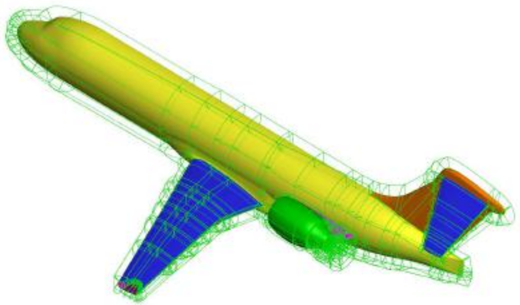
实际工程计算求解的控制方程 ≠ NS方程

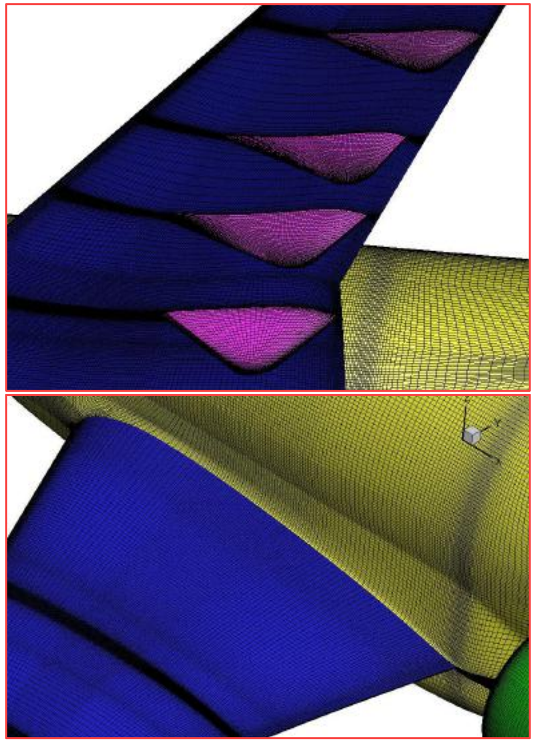
工程上常用的是雷诺时均N-S方程，和大涡模拟（滤波NS方程）

- Spalart预计： 2080年超级计算机可以直接求解NS方程
- 工程设计一般关心流动的统计结果（计算量依然很大）









本质上工程求解的控制方程都是不准确的

"Essentially, all models are wrong, but some are useful"

Your model *has to be* wrong...

... but that's o.k. if it's illuminating!

George E.P. Box

PART 02

低速流动的数值算法

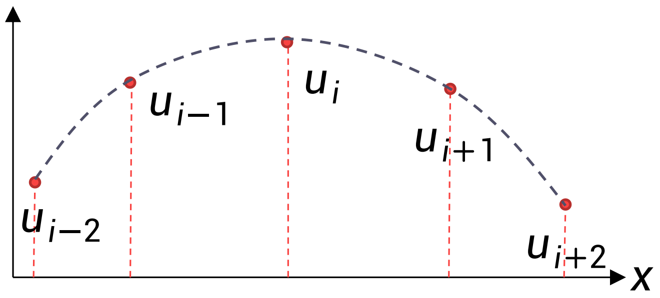
（相关数值算法以开源软件OpenFOAM为例说明）

不可压缩流动的Navier-Stokes方程

$$
\begin{cases}
\dfrac{\partial U_i}{\partial x_i} = 0 \\
\dfrac{\partial U_i}{\partial t} + U_j \dfrac{\partial U_i}{\partial x_j} = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x_i} + v\dfrac{\partial^2 U_i}{\partial x_j \partial x_j}
\end{cases}
$$

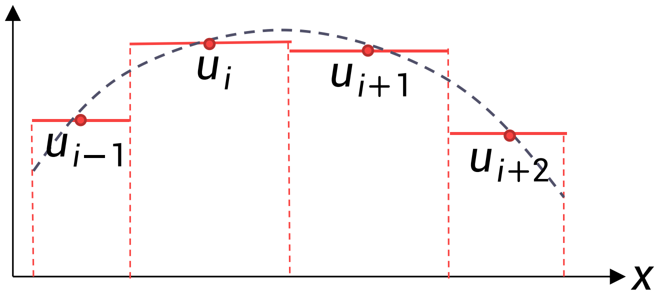
常见的离散格式

$$minimize\, R = \sum_{i=0}^{i=N} (u_i^h - u_i)^2$$

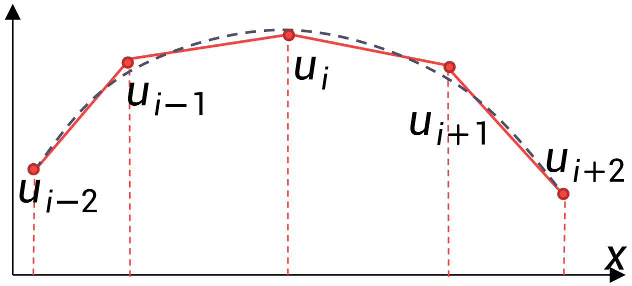


有限差分法

$$minimize\, R = \int_{x_{min}}^{x_{max}} w(x)u(x)dx$$

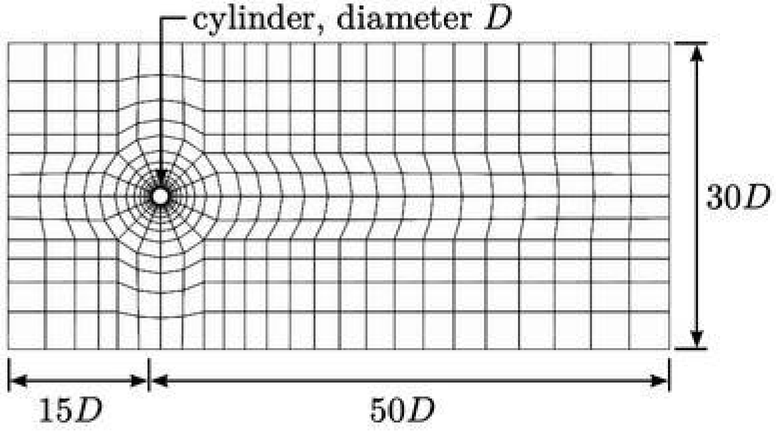


有限体积法

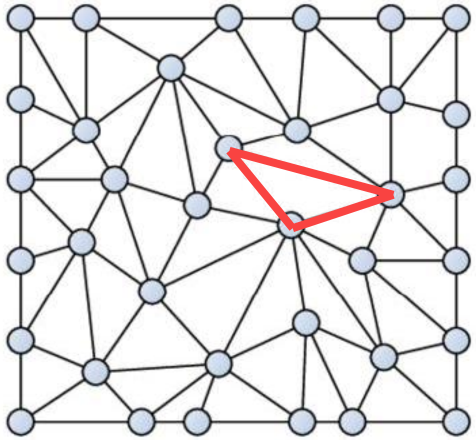


有限元方法

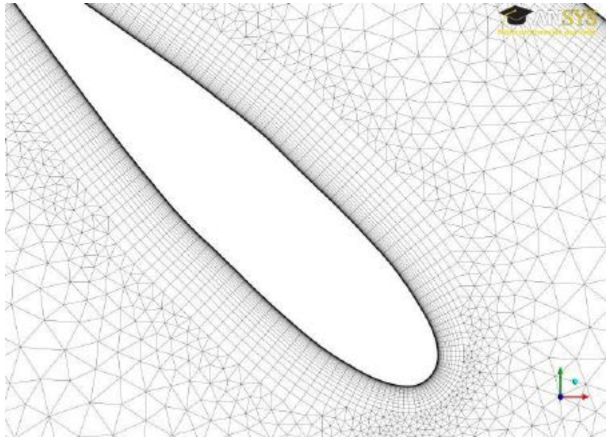
常见的离散手段



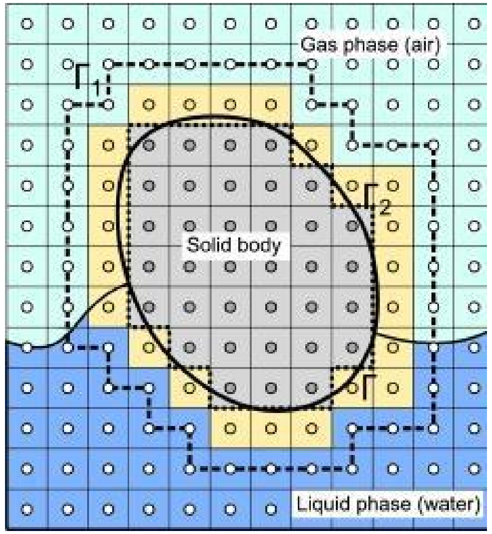
结构网格



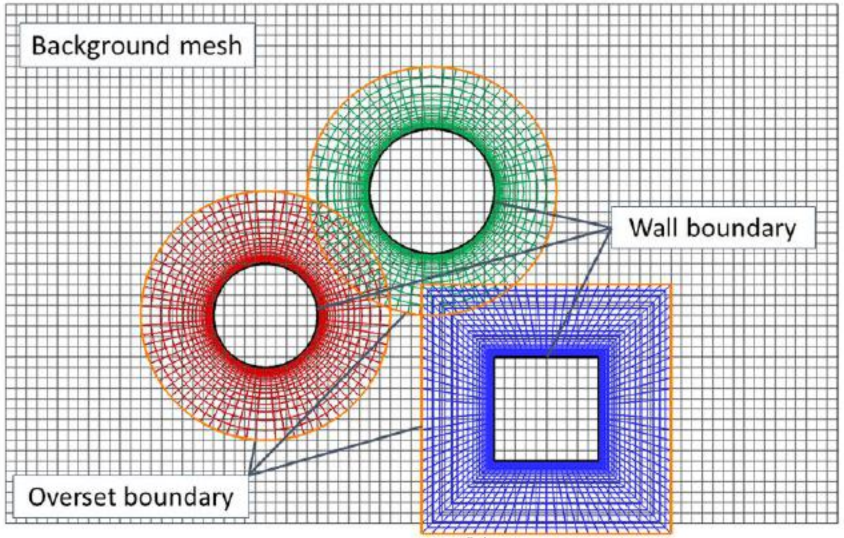
非结构网格



混合网格



浸没边界



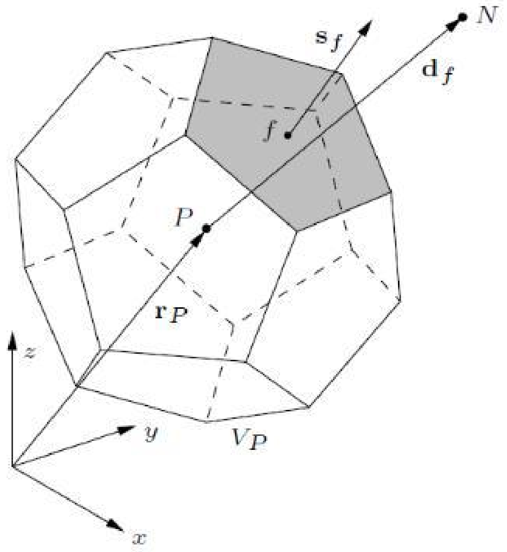
重叠网格

有限体积法基本原理

➢ **高斯定理**：任意体积分转化为封闭曲面积分

$$\int_V \nabla\cdot\mathbf{F}dV=\oint_S \mathbf{n}\cdot\mathbf{F}dS$$

其中，F为任意矢量， V是空间体积， S是空间 V的表面，*n*为 dS的单位法向量



➢ **标量场的梯度**：

对任意标量场T，计算梯度∇*T*

$$\int_V \nabla\cdot(\mathbf{c}T)dV=\int_V \mathbf{c}\cdot\nabla TdV$$

$$\rightarrow \int_V \nabla\cdot(\mathbf{c}T)dV=\oint_S \mathbf{n}\cdot(\mathbf{c}T)dS=\mathbf{c}\cdot\oint_S \mathbf{n}TdS$$

$$\rightarrow \int_V \mathbf{c}\cdot\nabla TdV=\mathbf{c}\cdot\int_V \nabla TdV$$

$$\rightarrow \mathbf{c}\cdot\int_V \nabla TdV=\mathbf{c}\cdot\oint_S \mathbf{n}TdS$$

得到标量梯度计算公式 $\int_V \nabla TdV=\oint_S \mathbf{n}TdS=\oint_S d\mathbf{S}T$

有限体积法基本原理

- **矢量场的梯度：**

对任意矢量场U，计算梯度∇U （或$\nabla\otimes U$，⊗为张量积符号）

$$a\otimes b \equiv a\cdot b^T$$

于是 ∇⊗U可以表示为

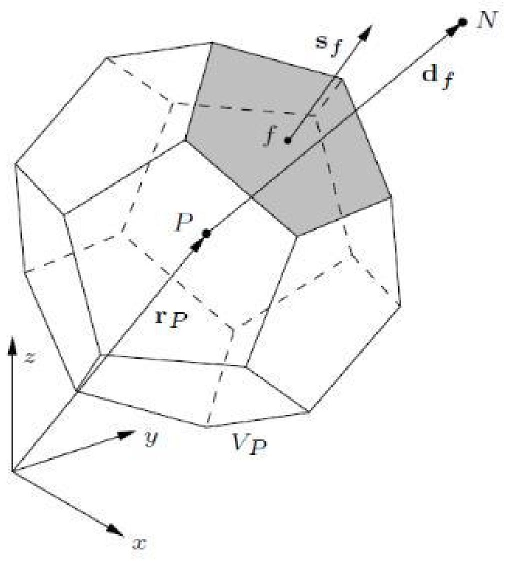
$$\nabla\otimes \mathrm{U} = \nabla\cdot U^T = (\nabla U_1; \nabla U_2; \nabla U_3)$$

对其积分，可得

$$\int_V \nabla\otimes\mathbf{U}dV = \int_V (\nabla U_1;\nabla U_2;\nabla U_3)dV = \oint_S (\mathbf{n}U_1;\mathbf{n}U_2;\mathbf{n}U_3)dS = \oint_S \mathbf{n}\otimes\mathbf{U}dS$$

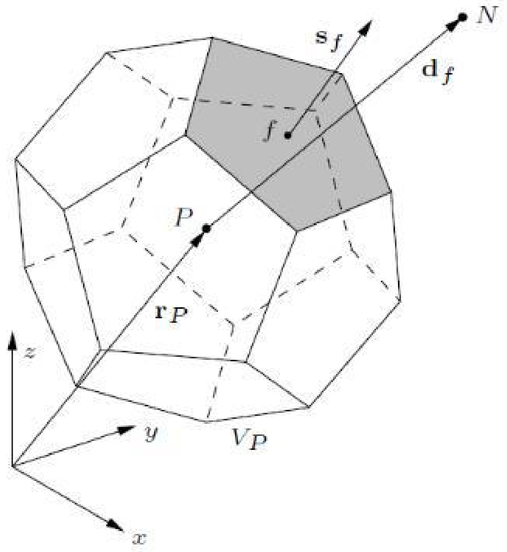
化简后得到

$$\int_V \nabla\otimes\mathbf{U}dV = \oint_S \mathbf{n}\otimes\mathbf{U}dS = \oint_S d\mathbf{S}\otimes\mathbf{U}$$



- **对于有限体积法的控制体，如何进一步计算梯度在体心（P）的值?**

有限体积法体心梯度的离散计算方法



- 首先计算体心位置：

$$\int_V (\mathbf{x}-\mathbf{x}_P)dV = 0$$

- 假定场量ϕ在控制体单元内**线性变化**，控制体内任意一点x的ϕ_x满足：

$$\phi_{\mathbf{x}} \approx \phi_P + (\nabla\phi)_P \cdot (\mathbf{x}-\mathbf{x}_P)$$

- 对控制体进行积分

$$\int_V \phi_{\mathbf{x}} dV = \int_V [\phi_P + (\nabla\phi)_P \cdot (\mathbf{x}-\mathbf{x}_P)]dV = \phi_P V$$

- 上式表示在控制体内线性变化的物理量 ϕ 对控制体积分等于体心的值ϕ_P乘以控制体体积 V，即

$$\phi_P = \frac{1}{V}\int_V \phi_{\mathrm{x}} dV$$

- 因此，标量T在体心处的梯度可以表示为

$$(\nabla T)_P = \frac{1}{V}\int_V (\nabla T)dV = \frac{1}{V}\oint_S d\mathbf{S}T = \frac{1}{V}\sum_{\mathrm{face}} \mathbf{S}_{\mathrm{face}} T_{\mathrm{face}}$$

同理 $$(\nabla \otimes \mathbf{U})_P = \frac{1}{V}\sum_{\mathrm{face}} \mathbf{S}_{\mathrm{face}} \otimes \mathbf{U}_{\mathrm{face}}$$

对流扩散方程的离散

- 考虑一般形式的对流扩散方程，其形式可以表示如下

$$\underbrace{\frac{\partial \phi}{\partial t}}_{\text{时间项}} + \underbrace{\nabla \cdot (\phi \mathbf{u})}_{\text{对流项}} - \underbrace{\nabla \cdot (\gamma \nabla \phi)}_{\text{扩散项或站性项}} = \underbrace{S_\phi(\phi)}_{\text{源项}}$$

- 对单元做体积分，可得

$$\int_{V_P} \frac{\partial \phi}{\partial t} dV + \int_{V_P} \nabla \cdot (\phi \mathbf{u}) dV - \int_{V_P} \nabla \cdot (\gamma \nabla \phi) dV = \int_{V_P} S_\phi(\phi) dV$$

- 通过高斯定理将对流项和扩散项的提积分转为面积分，则

$$\int_{V_P} \frac{\partial \phi}{\partial t} dV + \sum_f [\mathbf{S}_f \cdot (\phi \mathbf{u})_f] - \sum_f [\mathbf{S}_f \cdot (\gamma \nabla \phi)_f] = \int_{V_P} S_\phi(\phi) dV$$

- 首先处理时间项

$$\int_{V_P} \frac{\partial \phi}{\partial t} dV = \frac{\phi^n - \phi^o}{\Delta t} V_P$$

对流扩散方程的离散

- 考虑一般形式的对流扩散方程，其形式可以表示如下

$$\underbrace{\frac{\partial \phi}{\partial t}}_{\text{时间项}} + \underbrace{\nabla \cdot (\phi \mathbf{u})}_{\text{对流项}} - \underbrace{\nabla \cdot (\gamma \nabla \phi)}_{\text{扩散项或站性项}} = \underbrace{S_\phi(\phi)}_{\text{源项}}$$

- 对流项的处理：面心物理量通过体心插值得到

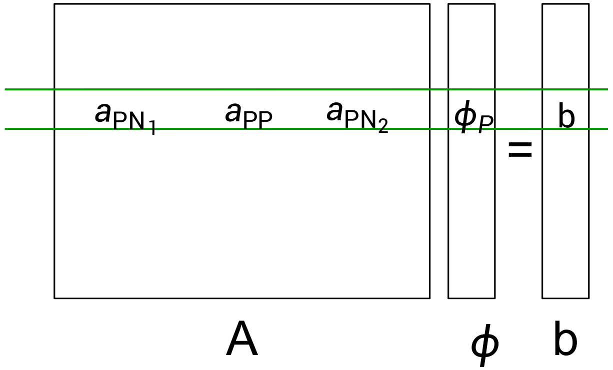
$$\phi_f = \lambda \phi_P + (1-\lambda)\phi_N \qquad \lambda = \overline{fN}/\overline{PN}$$

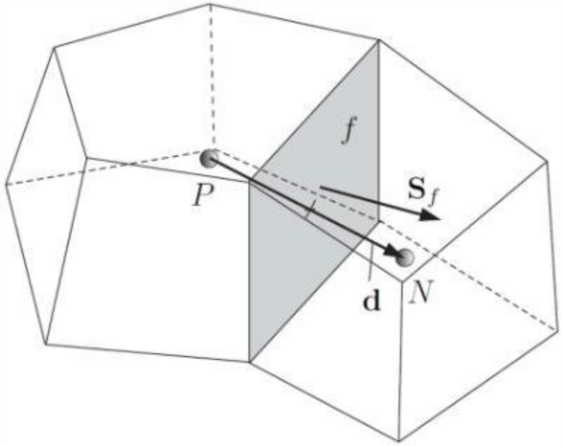
- 对流项可表示为

$$\sum_f [\mathbf{S}_f \cdot (\phi \mathbf{u})_f] = \underbrace{\sum_f [\lambda(\mathbf{S}_f \cdot \mathbf{u}_P)]\phi_P}_{\text{P 单元}} + \underbrace{\sum_f [(1-\lambda)(\mathbf{S}_f \cdot \mathbf{u}_N)\phi_N]}_{\text{与 P 相邻的所有单元}}$$

- 其中ϕ_P、 ϕ_N 是未知数

- 系数填入线性方程组





f 代表面单元上的物理量

S_f 代表面法向向量

$|S_f|$ 代表面的面积

V_P为控制体P的体积

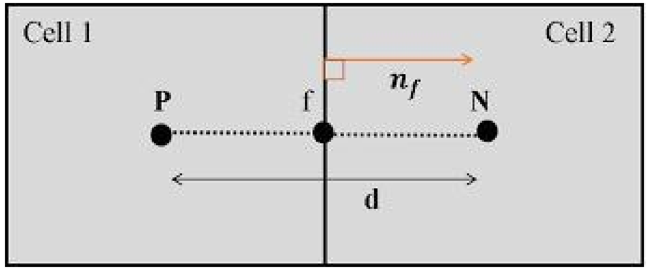
V_N为控制体N的体积

对流扩散方程的实现过程

对流扩散方程的离散

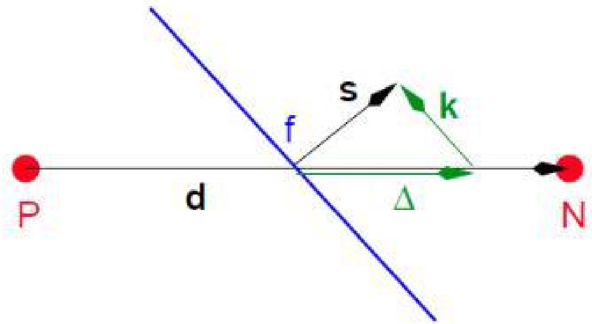
- 考虑一般形式的对流扩散方程，其形式可以表示如下

$$\underbrace{\frac{\partial \phi}{\partial t}}_{\text{时间项}} + \underbrace{\nabla \cdot (\phi \mathbf{u})}_{\text{对流项}} - \underbrace{\nabla \cdot (\gamma \nabla \phi)}_{\text{扩散项或站性项}} = \underbrace{S_\phi(\phi)}_{\text{源项}}$$



- 扩散项的处理：
  - 只考虑正交网格：

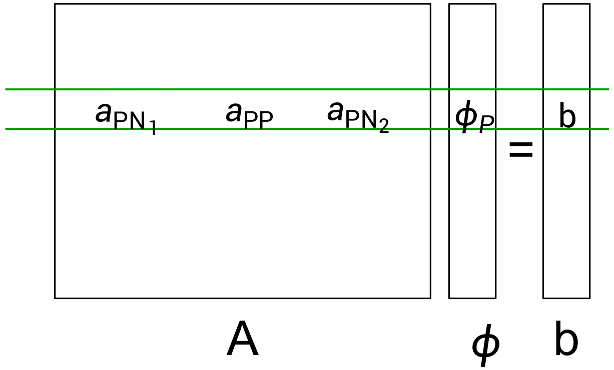
$$-\sum_f [\mathbf{S}_f \cdot (\gamma \nabla \phi)_f] = -\sum_f \gamma_f |\mathbf{S}_f| \frac{\phi_N - \phi_P}{|PN|}$$



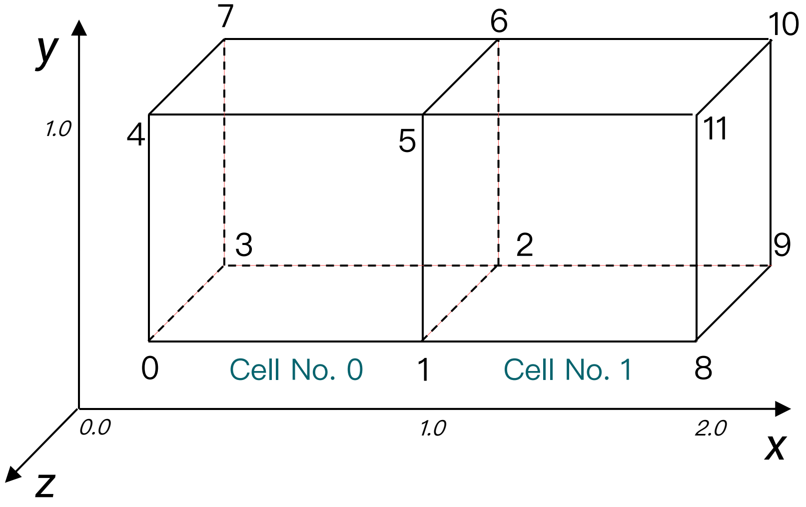
  - 考虑非正交网格：

$$\mathbf{S}_f \cdot (\nabla \phi)_f = \underbrace{|\Delta| \frac{\phi_N - \phi_P}{|PN|}}_{\text{正交部分}} + \underbrace{\mathbf{k} \cdot (\nabla \phi)_f}_{\text{非正交修正部分}}$$

  - 正交部分作为左端A系数，非正交修正给出右端b



对流扩散方程的离散



Cell No. 0	0	1	2	3	4	5	6	7
Cell No. 1	1	2	9	8	5	11	10	6

控制体数组存储的是各控制体对应的顶点编号

Face ID					Owner ID	Neighbor ID
0	1	2	6	5	0	1
1	0	1	5	4	0	NULL
2	0	3	7	4	0	NULL
3	0	1	2	3	0	NULL
4	2	3	7	6	0	NULL
5	4	5	6	7	0	NULL
6	1	8	9	2	1	NULL
7	1	8	11	5	1	NULL
8	5	11	10	6	1	NULL
9	2	9	10	6	1	NULL
10	8	9	10	11	1	NULL

面的数组存储的是组成面的顶点编号

Owner存储的是面一侧的控制体编号

Neighbor存储的是面另一侧的顶点编号

Node ID	0	1	2	3	4	5	6	7	8	9	10	11
x coordinate	0	1	1	0	0	1	1	0	2	2	2	2
y coordinate	0	0	0	0	1	1	1	0	0	0	1	1
z coordinate	0	0	1	1	0	0	1	1	0	–1	–1	0

顶点数组存储的是每个顶点的坐标信息

NS方程的离散过程

在完成一般对流扩散方程的离散…　　怎么把NS方程离散?

- **不可压缩NS方程：**

$$\begin{cases} \dfrac{\partial U_i}{\partial x_i} = 0 \\ \dfrac{\partial U_i}{\partial t} + U_j \dfrac{\partial U_i}{\partial x_j} = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x_i} + v\dfrac{\partial^2 U_i}{\partial x_j \partial x_j} \end{cases}$$

- **动量方程：**

$$\underbrace{\frac{\partial \phi}{\partial t}}_{\text{时间项}} + \underbrace{\nabla\cdot(\phi\mathbf{u})}_{\text{对流项}} - \underbrace{\nabla\cdot(\gamma\nabla\phi)}_{\text{扩散项或站性项}} = \underbrace{S_\phi(\phi)}_{\text{源项}} \Rightarrow \frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} - \frac{\partial}{\partial x_j}\left(v\frac{\partial U_i}{\partial x_j}\right) = -\frac{1}{\rho}\frac{\partial P}{\partial x_i}$$

- **连续性方程：**

$$\frac{\partial U_i}{\partial x_i} = 0 \Rightarrow$$ U_i已经有输运方程了，如何耦合?

- **SIMPLE：** Semi–implicit Method for Pressure Linked Equations (Patankar & Spalding)

NS方程的离散过程

在完成一般对流扩散方程的离散…　　　　那么怎么把NS方程离散?

➢ **不可压缩NS方程（定常举例）：**

$$
\begin{cases}
\dfrac{\partial U_i}{\partial x_i} = 0 \\
U_j \dfrac{\partial U_i}{\partial x_j} - \dfrac{\partial}{\partial x_j}\left(v \dfrac{\partial U_i}{\partial x_j}\right) = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x_i}
\end{cases}
$$

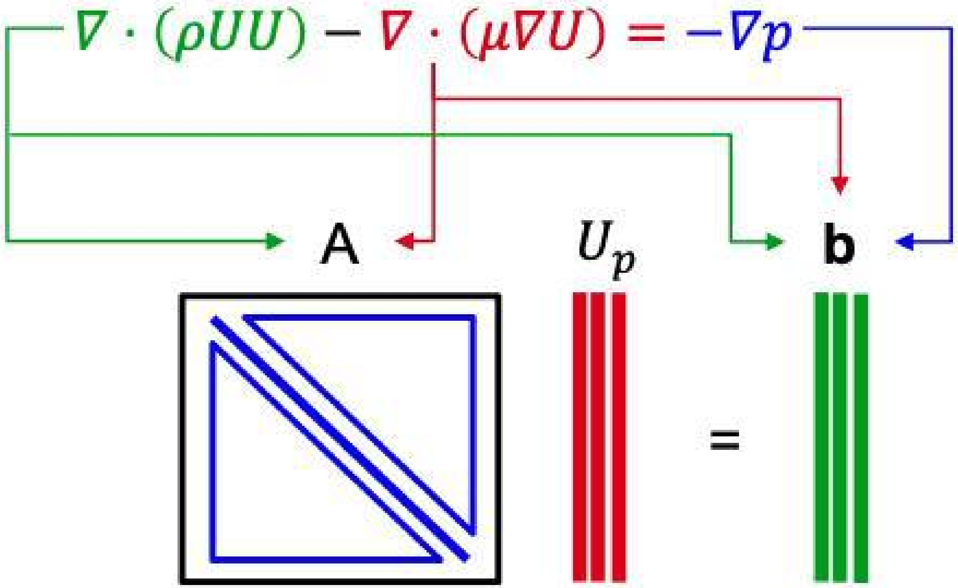
➢ **第一步：组装动量方程的线性方程组**

➢ 对流项离散：

$$\sum_f [\mathbf{S}_f \cdot (\phi \mathbf{u})_f] = \underbrace{\sum_f [\lambda(\mathbf{S}_f \cdot \mathbf{u}_P)]\phi_P}_{\text{P 单元}} + \underbrace{\sum_f [(1-\lambda)(\mathbf{S}_f \cdot \mathbf{u}_N)\phi_N]}_{\text{与 P 相邻的所有单元}}$$

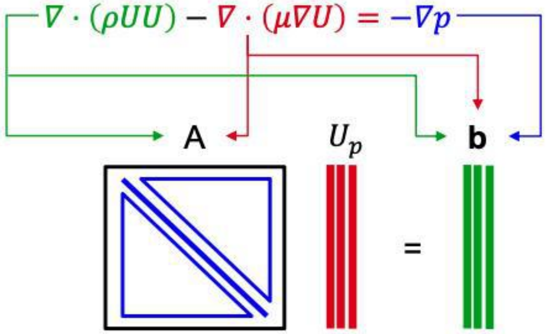
➢ 扩散项离散：

$$-\sum_f [\mathbf{S}_f \cdot (\gamma \nabla \phi)_f] = -\sum_f \gamma_f |\mathbf{S}_f| \frac{\phi_N - \phi_P}{|\overline{PN}|}$$



不可压缩流动的SIMPLE算法

通过解耦的方式，分别求解压强和动量方程，来满足质量守恒和动量守恒



- **第二步：根据 $AU_p = b$求解U_p**
  - 迭代法求解线性方程组：PBiCG、PBiCGStab等
- **第三步：重写动量线性方程组**
  - $AU_p = (L+D+U)U_p = \tilde{b} = b + \nabla p$

    ⇩

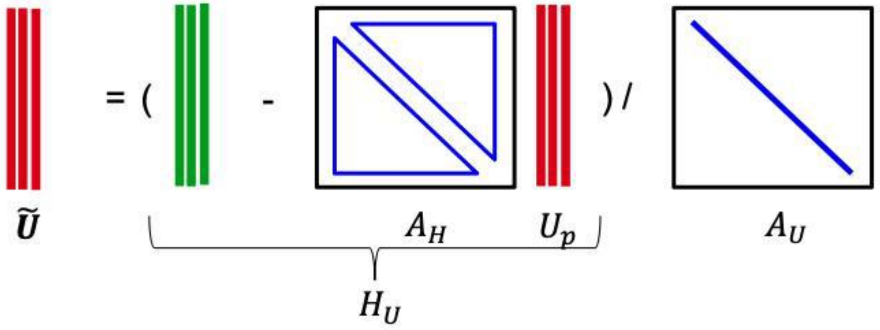
  - $DU_p = \tilde{b} - (L+U)U_p$

    ⇩

  - $U_p = [\tilde{b} - (L+U)U_p]/D$

    ⇩

  - 记为 $\tilde{U} = H_U/A_U$ 其中 $H_U = [\tilde{b} - (L+U)U_p]$， $A_U = D$



$\tilde{U}$不包含压强的影响

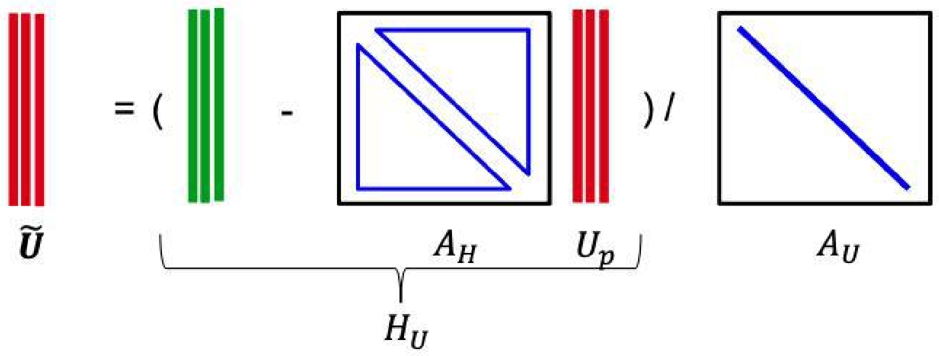
注：正交网格下$\tilde{b}=0$

不可压缩流动的SIMPLE算法

通过解耦的方式，分别求解压强和动量方程，来满足质量守恒和动量守恒

- **第三步：重写动量线性方程组**

$$\tilde{U} = H_U/A_U \qquad \text{其中 } H_U = [\tilde{b} - (L+U)U_p],\quad A_U = D$$



- **第四步**：求解压强泊松方程

$$\sum_f \left(\frac{\rho}{A_U}\right)_{\mathrm{f}} (\nabla \mathbf{p})_{\mathrm{f}} \cdot S = \sum_f \rho \tilde{U} \cdot S$$

- **第五步**：更新质量通量

$$F = \rho U_f \cdot S = \rho \tilde{U} \cdot S - \left(\frac{\rho}{A_U}\right)_{\mathrm{f}} (\nabla \mathbf{p})_{\mathrm{f}} \cdot S$$

- **已完成一次SIMPLE迭代，循环执行所有步骤直到**$\sum_f F = 0$
- **问题：**
  - 连续性方程去哪了？什么是$\sum_f F = 0$的物理含义？

SIMPLE方法通过联立连续性方程和动量方程，得到压强方程

➢ 根据动量方程的离散形式

$$AU_p = (L + D + U)U_p = -\nabla p$$

令$D = a_P$，$H_U = -(L + U)U_P$

➢ 那么速度可以写成

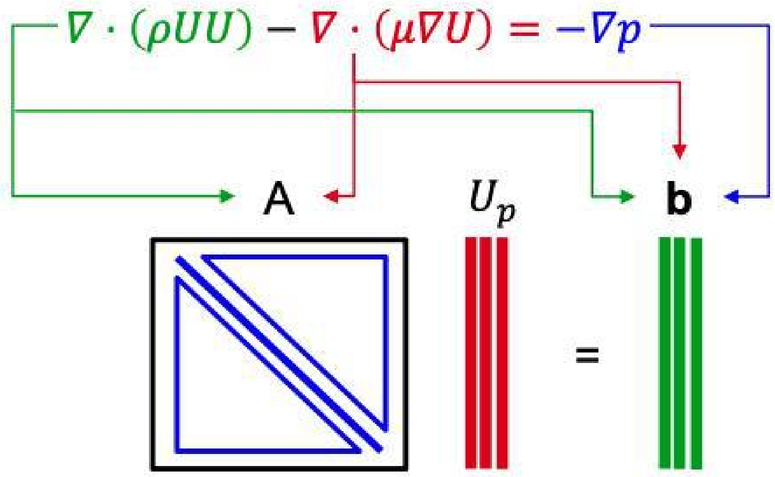
$$a_P U_p = H_U - \nabla p \qquad U_p = \frac{H_U}{a_P} - \frac{1}{a_P}\nabla p$$

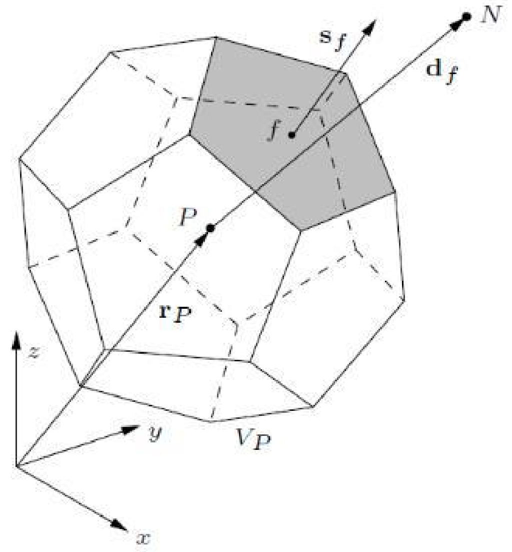
➢ 通过插值到面上

$$U_f = \left(\frac{H_U}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f$$

➢ 引入连续性方程

$$\nabla \cdot U = \sum_f S \cdot U_f = 0$$





不可压缩流动的SIMPLE算法

SIMPLE方法通过联立连续性方程和动量方程，得到压强方程

➢ 通过插值到面上并结合连续性方程

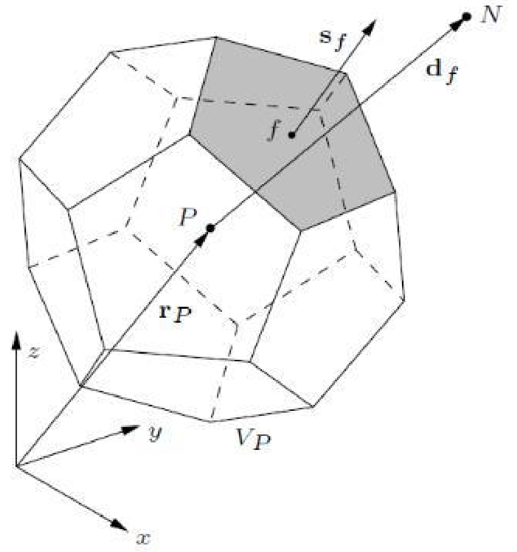
$$\boldsymbol{U}_f = \left(\frac{\boldsymbol{H}_U}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla \mathrm{p})_f \qquad \nabla \cdot \mathbf{U} = \sum_f \mathbf{S} \cdot \boldsymbol{U}_f = 0$$

➢ 通过插值到面上并结合连续性方程

$$\sum_f \mathbf{S} \cdot \boldsymbol{U}_f = \sum_f \mathbf{S} \cdot \left[\left(\frac{\boldsymbol{H}_U}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla \mathrm{p})_f\right] = 0 \quad \Rightarrow \quad \sum_f \mathbf{S} \cdot \left[\left(\frac{1}{a_P}\right)_f (\nabla \mathrm{p})_f\right] = \sum_f \mathbf{S} \cdot \left[\left(\frac{\boldsymbol{H}_U}{a_P}\right)_f\right]$$

➢ 得到压强泊松方程（本质上就是求解连续性方程）

$$\nabla \cdot \left(\frac{1}{a_p} \nabla \mathrm{p}\right) = \sum_f S \cdot \left[\left(\frac{H_U}{a_P}\right)_f\right]$$



➢ 每个面上的质量通量可以写作

$$\mathbf{S} \cdot \boldsymbol{U}_f = \mathbf{S} \cdot \left[\left(\frac{\boldsymbol{H}_U}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla \mathrm{p})_f\right] = 0$$

SIMPLE方法将动量方程和压强泊松方程转化为线性方程组

- 压强方程一般是对称的——共轭梯度法

$$
\begin{aligned}
&\textbf{input } A, b, x_0 \\
&r_0 = b - Ax_0 \\
&p_0 = r_0 \\
&\textbf{for } k = 1, \ldots \text{ until convergence } \textbf{do} \\
&\quad \gamma_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}} \\
&\quad x_k = x_{k-1} + \gamma_{k-1} p_{k-1} \\
&\quad r_k = r_{k-1} - \gamma_{k-1} A p_{k-1} \\
&\quad \delta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}} \\
&\quad p_k = r_k + \delta_k p_{k-1} \\
&\textbf{end for}
\end{aligned}
$$

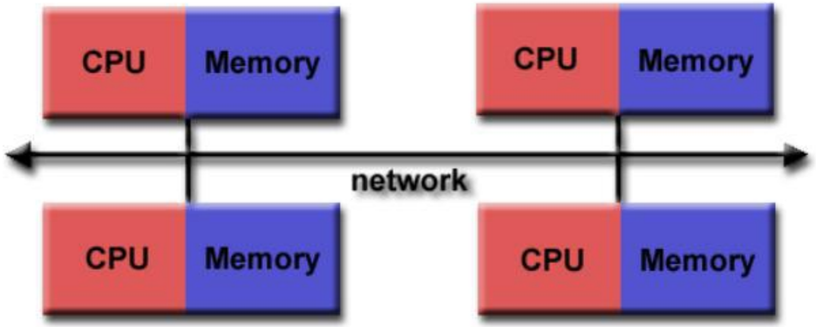
- 对于非对称的动量方程，可以采用BiCG、BiCGStab、GMRES等算法
- 此外对于大规模计算，还可以采用GAMG多重网格方法

PART 03

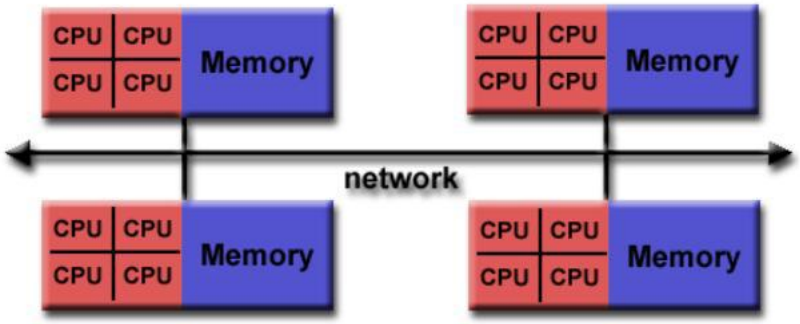
MPI并行计算开发

MPI是什么：跨节点、节点内数据发送接受的标准接口

➢ 基本的MPI编程模式

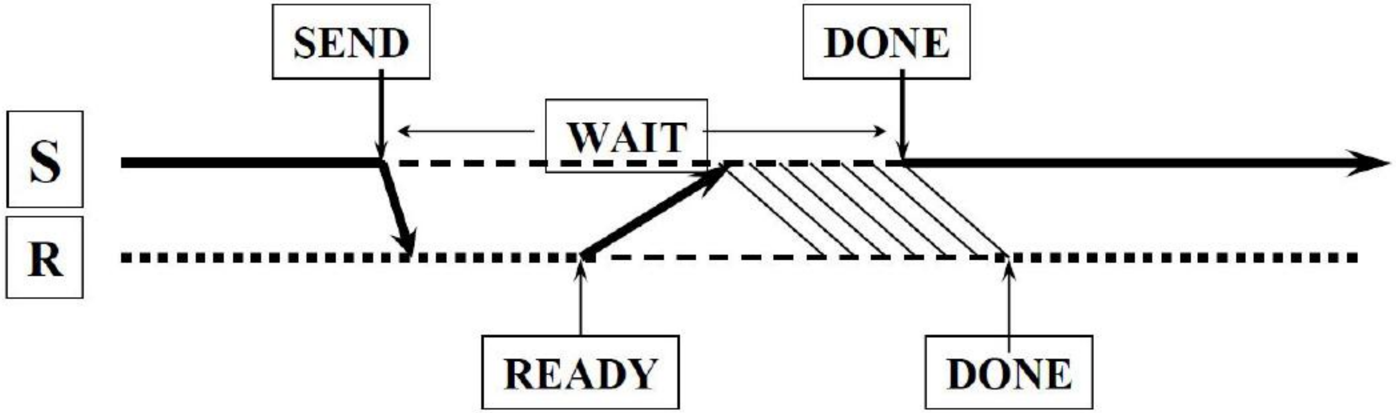


早期的MPI编程模式



现代的MPI编程模式

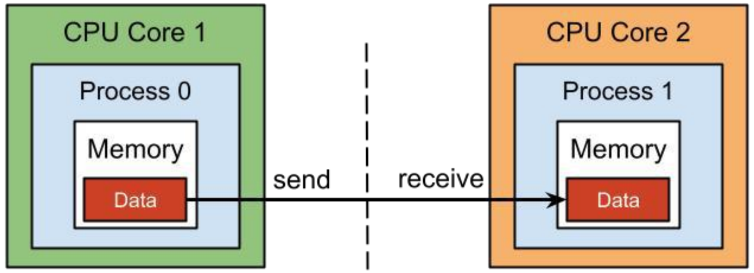
➢ 基本的通信模式（例子：阻塞式点对点通信）



MPI通信方法

MPI常用的通信模式有点对点通信和收集通信

- **点对点通信**



点对点通信

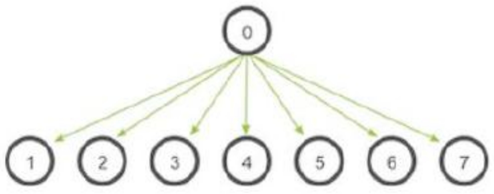
```
MPI_Send(
    void*           data,
    int             count,
    MPI_Datatype    datatype,
    int             destination,
    int             tag,
    MPI_Comm        communicator)
```

MPI send

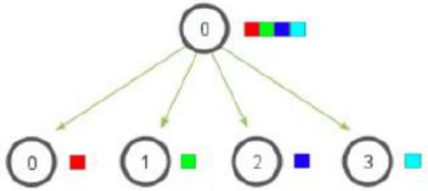
```
MPI_Recv(
    void*           data,
    int             count,
    MPI_Datatype    datatype,
    int             source,
    int             tag,
    MPI_Comm        communicator,
    MPI_Status*     status)
```

MPI recv

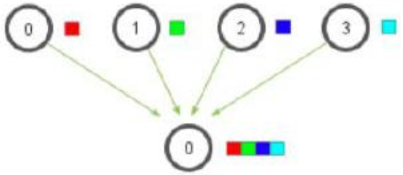
- **收集通信**



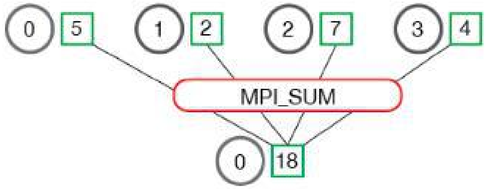
Broadcast



Scatter



Gather



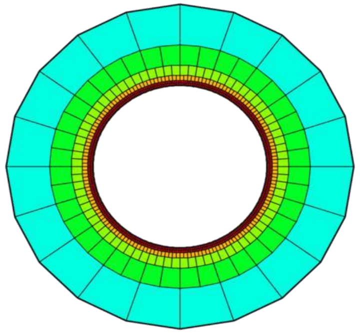
Reduce

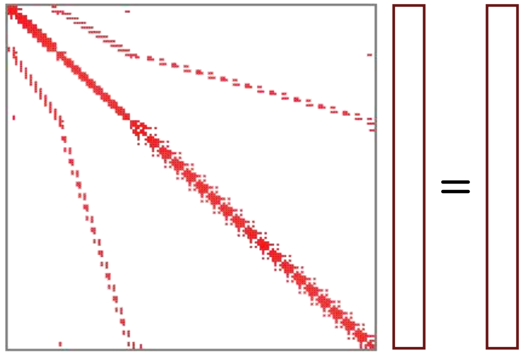
- **思考：哪些是计算流体力学中常用的通信方式?**

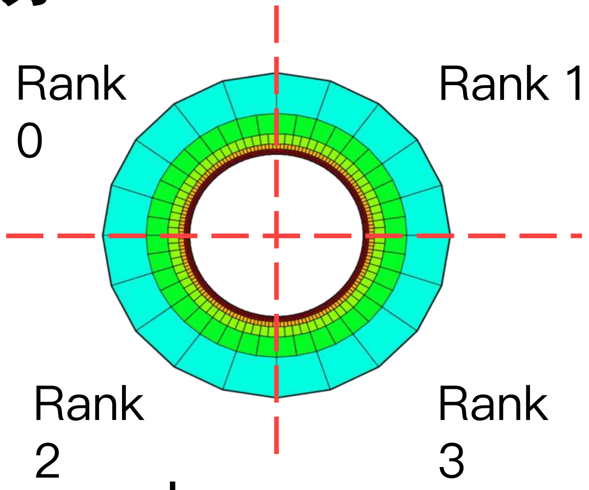
计算流体力学的基本步骤（回顾）

MPI可以帮助进行空间离散的并行化，不能解决时间步的拆分

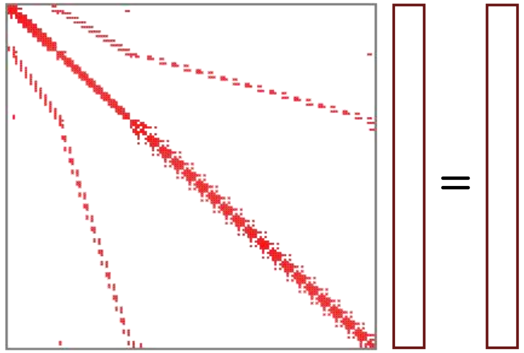
- **求解过程**
  - 步骤一：建立计算网格
  - 步骤二：初始化流场解
  - 步骤三：时间迭代
    - 组装动量线性方程组
    - 求解动量方程
    - 组装压强线性方程组
    - 求解压强方程
  - 步骤四：存储计算结果，后处理







得到什么样的线性方程组?

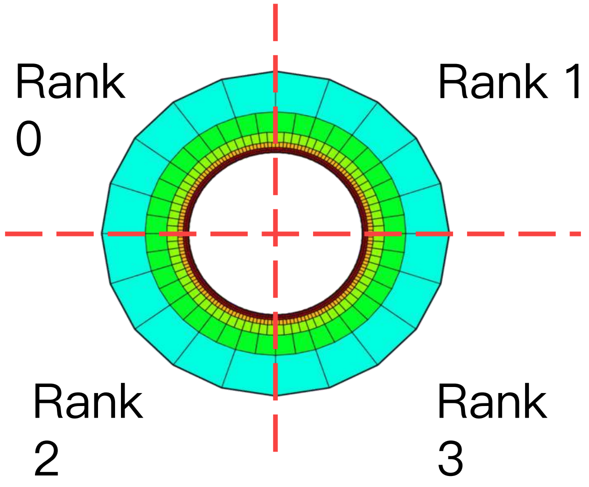


- **思考：空间离散的并行化如何实现?**
  - 如右图的拆分方式，会得到怎样的线性方程组?

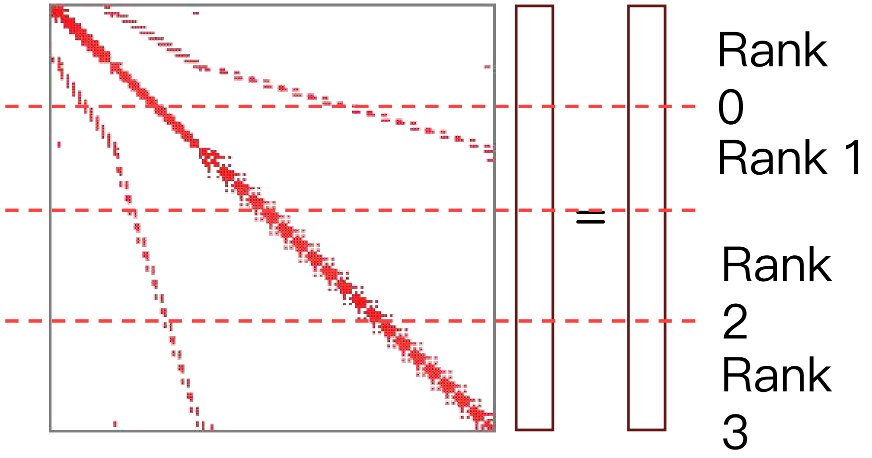
网格拆分需求

MPI可以帮助进行空间离散的并行化，不能解决时间步的拆分

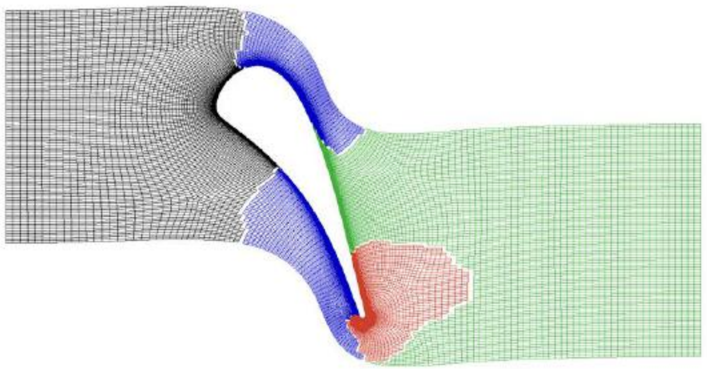
- 对于任意的非结构网格，编号顺序与空间无关



$\nRightarrow$



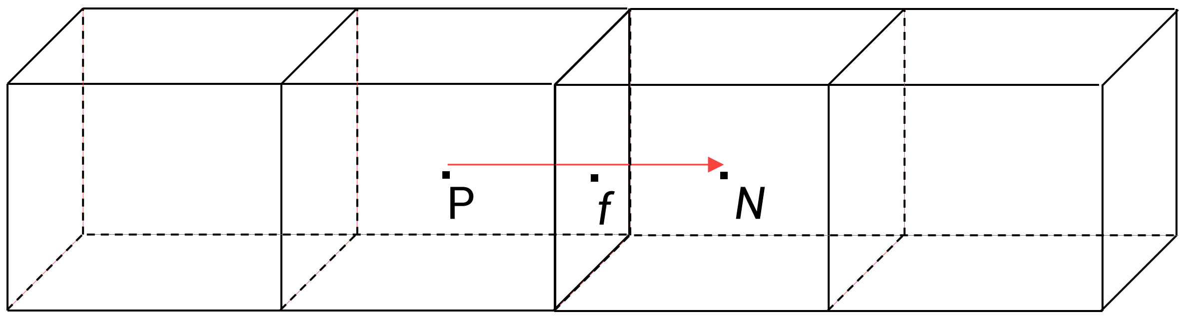
- 网格划分之后必须要进行重新编号
  - 前提：得到网格划分方案
  - 简单几何可以通过坐标定义
  - 复杂几何如何得到最优的划分方案?



网格划分和并行通讯之间的关系

- 以对流项离散举例：

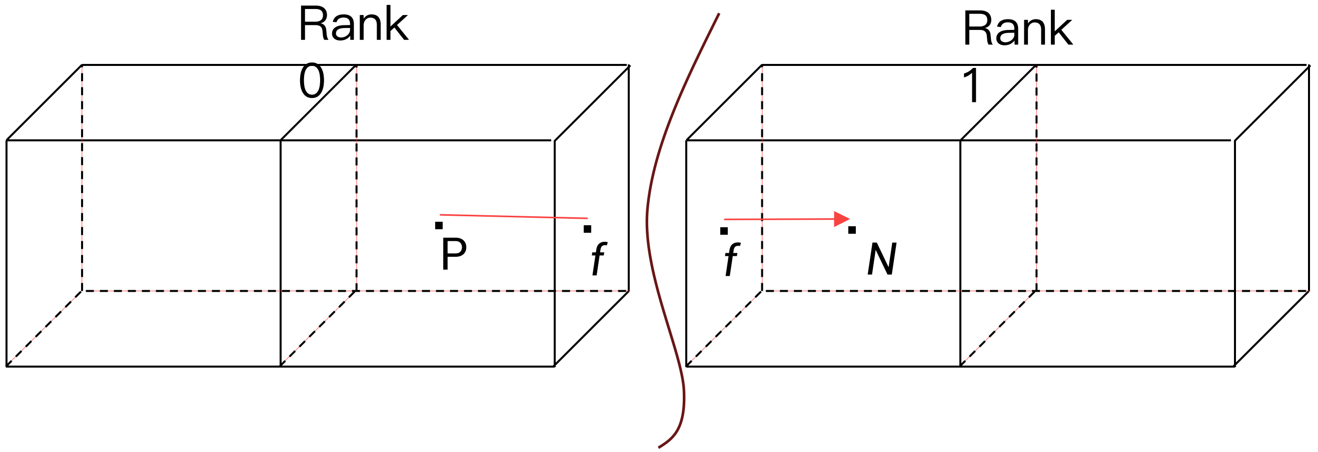
$$\sum_f [\mathbf{S}_f \cdot (\phi \mathbf{u})_f] = \underbrace{\sum_f [\lambda(\mathbf{S}_f \cdot \mathbf{u}_P)]\phi_P}_{\text{P 单元}} + \underbrace{\sum_f [(1-\lambda)(\mathbf{S}_f \cdot \mathbf{u}_N)\phi_N]}_{\text{与 P 相邻的所有单元}}$$



- 任意网格面：

$$S_f \cdot (u\phi)_f = \lambda S_f \cdot (u\phi_P) + (1-\lambda) S_f \cdot (u\phi_N)$$

- 为了并行计算，网格分块以后：
  - Rank 1需要把N的值传给Rank 0
  - Rank 0需要把P的值传给Rank 1



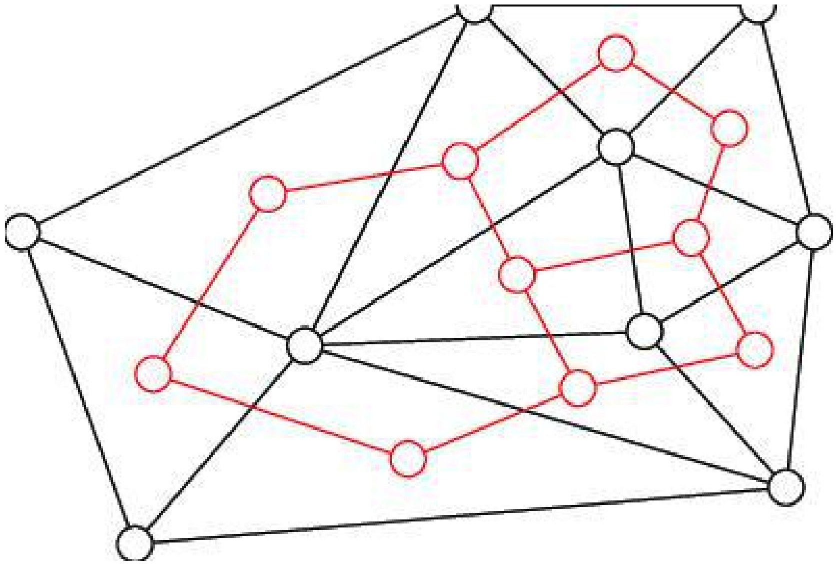
- **结论：通讯成本与被切开的面的数量成正比**

网格拆分需求

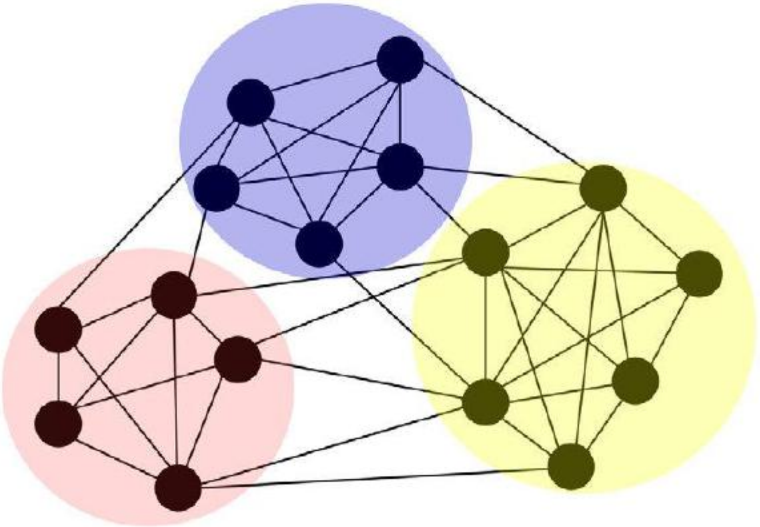
可以采用第三方工具帮助切分网格

- **基本原则：控制体之间相接的面被切开得越少越好!**

- 给定控制体和控制体之间的链接信息 → 转化为**图划分**（Graph Partitioning）问题



示意图：控制体、连接信息



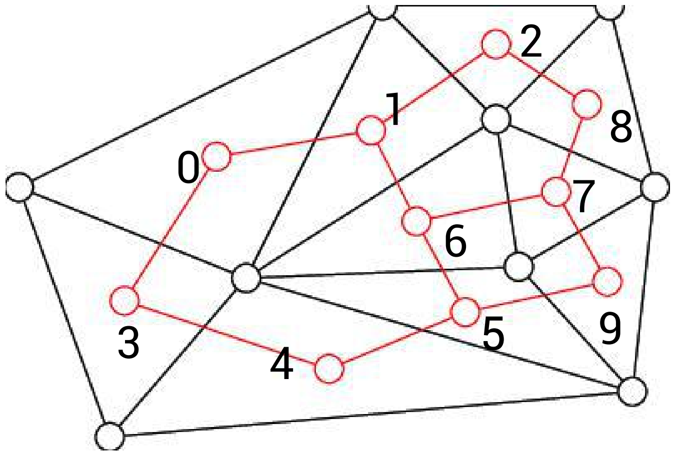
图划分问题示意图

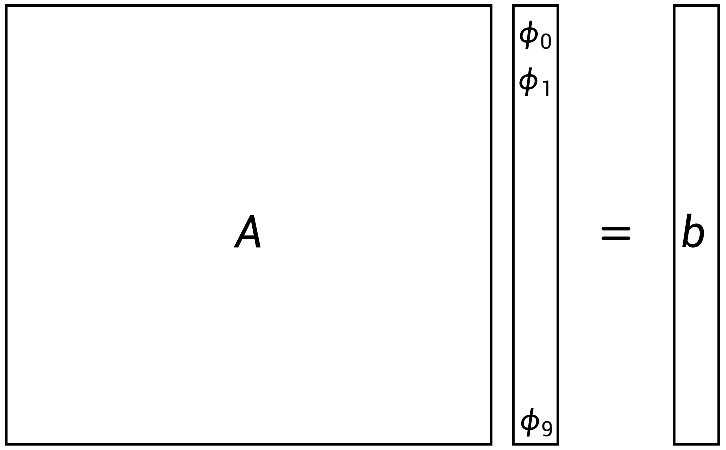
- **思考：切分的过程是否只需要网格点连接信息即可？需要网格点的具体坐标信息吗?**

网格重编号

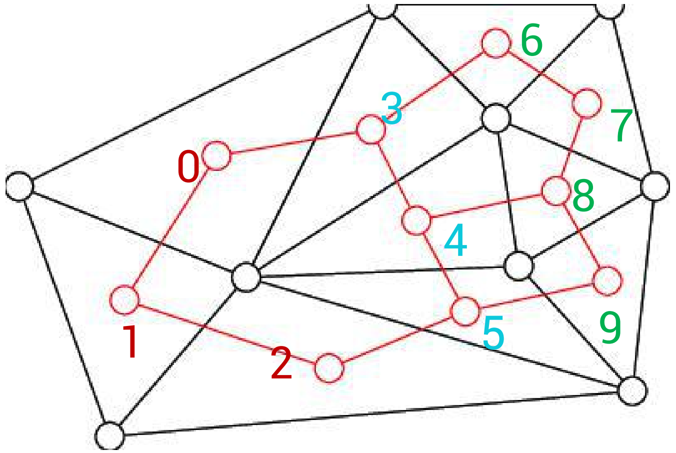
网格划分带来的是线性方程组重编号的需求

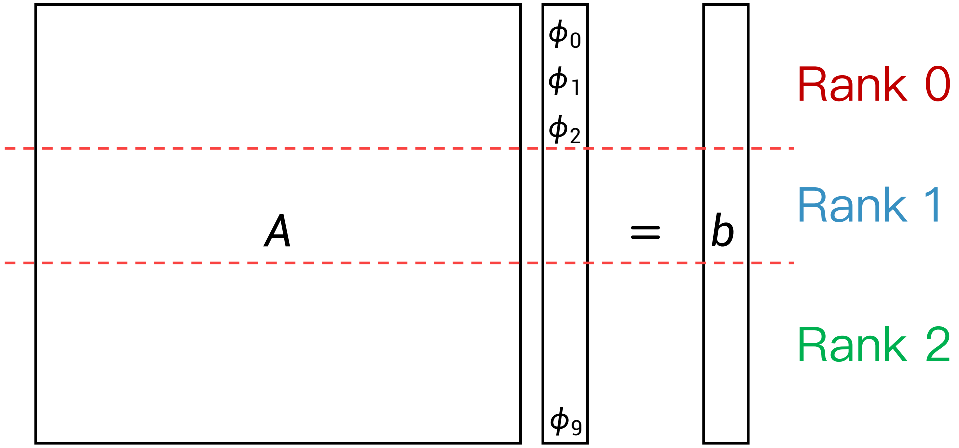
- 划分前的网格编号和线性方程组





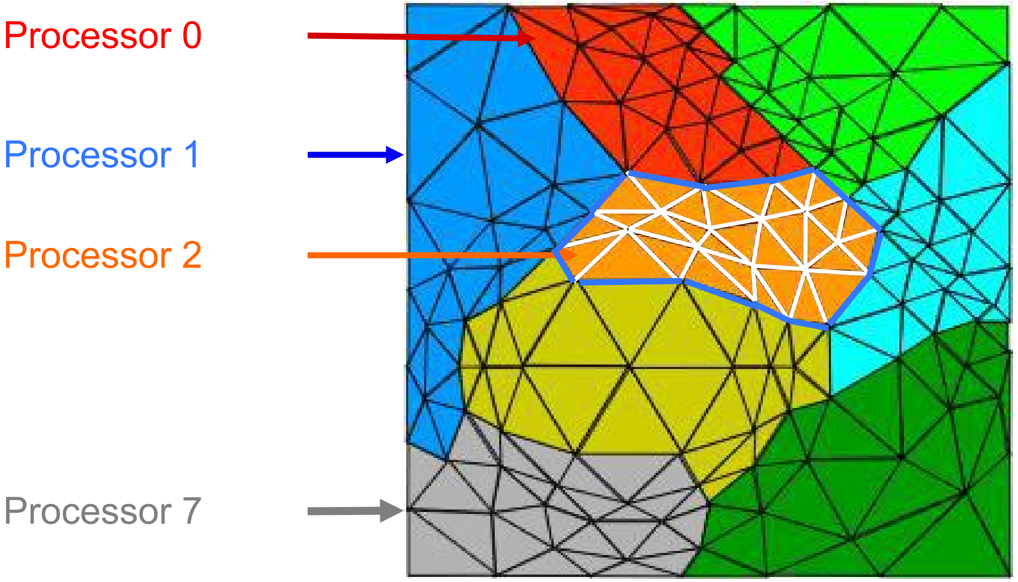
- 划分后的网格编号和线性方程组



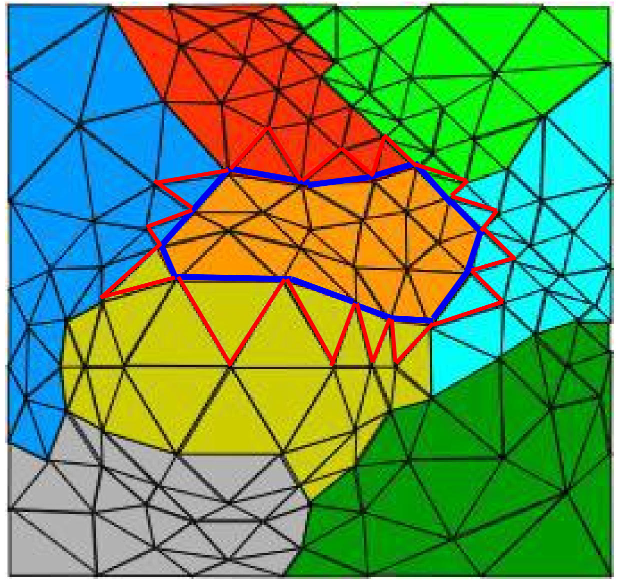


以Metis工具进行网格拆分的过程

- 以控制体标号的方式，将网格划分为多个处理器独立求解的控制体群组
- 对于非结构有限体积法，所需要的通信层数为一层



Sketch of mesh decomposition



Sketch of mesh decomposition

以Metis工具进行网格拆分的过程

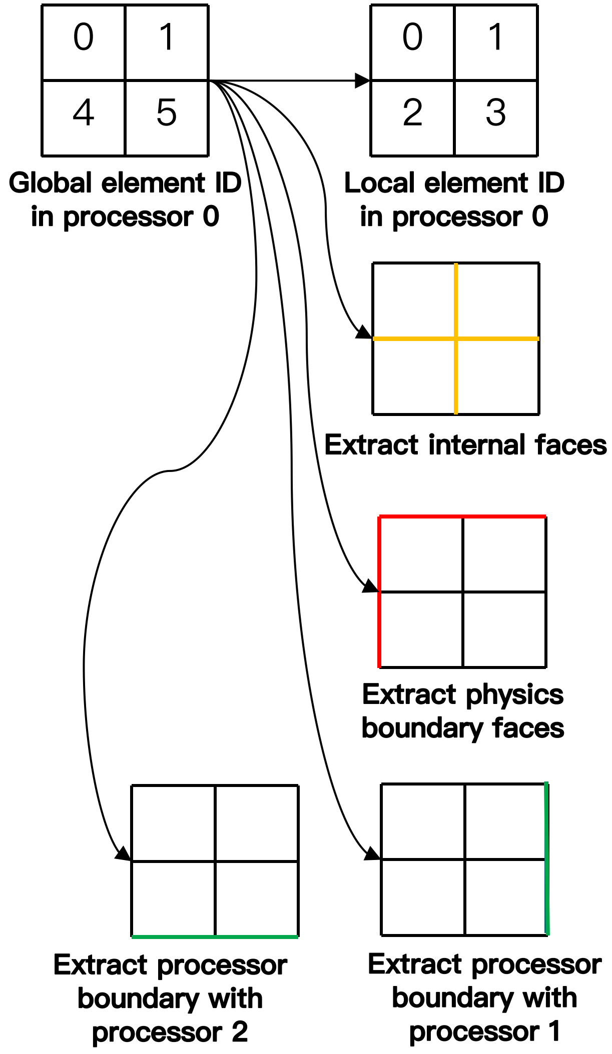
- Metis只负责返回每一个控制体对应的rank值
- 思考：实际MPI通信需要什么?
- 需要发送和接受数据的方向、数据量
- 需要额外的程序来识别通信面

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Element ID for a mesh

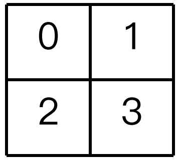
Element ID	Rank
0	0
1	0
2	1
3	1
4	0
5	0
6	1
7	1
8	2
9	2
10	3
11	3
12	2
13	2
14	3
15	3

METIS assigned rank

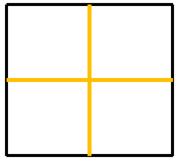


步骤一 ⇨ 步骤二 ⇨ 步骤三

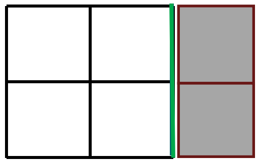
在MPI_Comm_World体系下建立数据结构和线性方程组求解方法



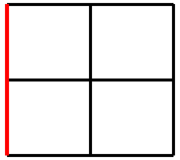
处理器上存在的控制体



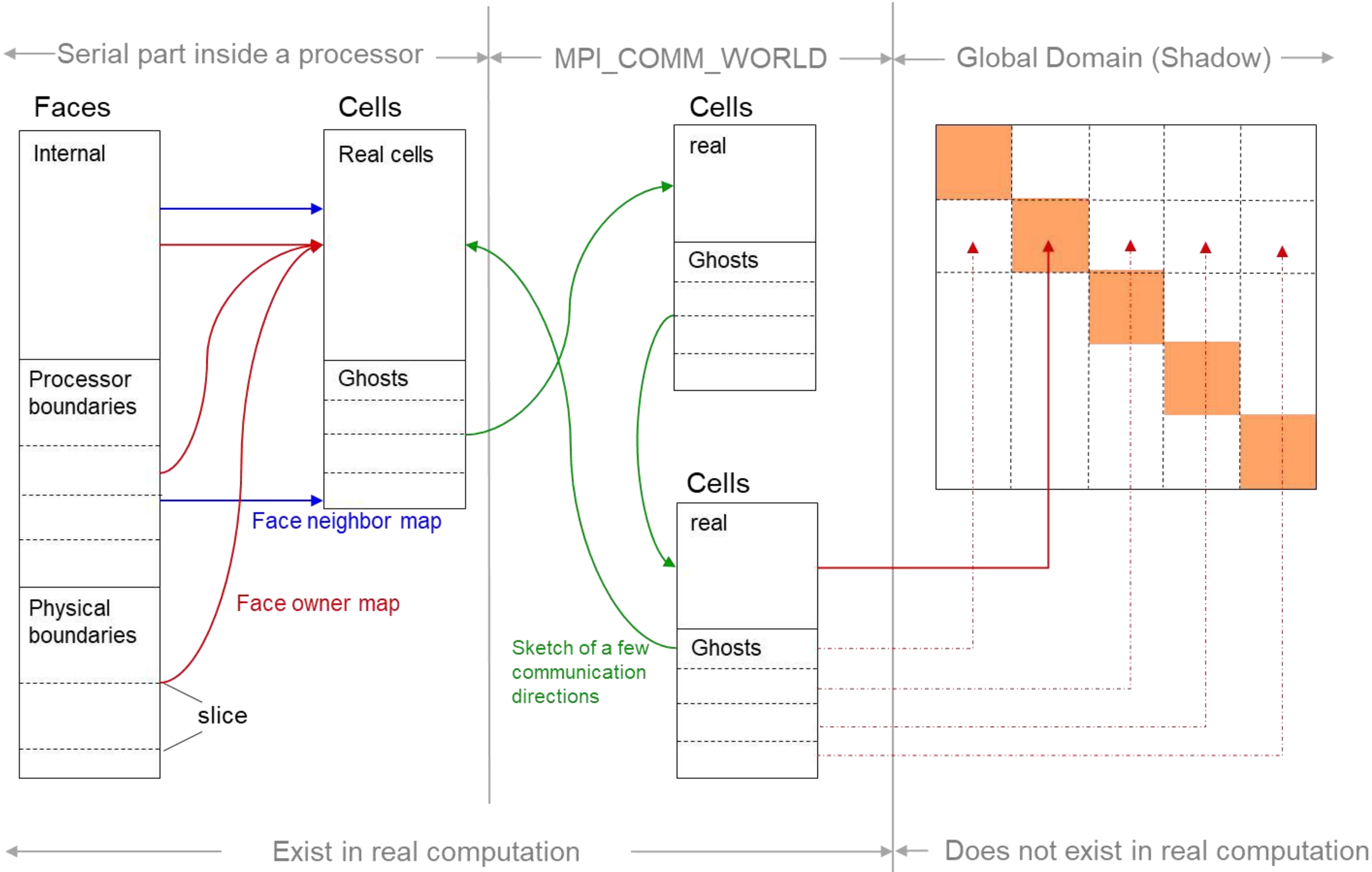
内部面



通讯边界



物理边界



线性方程组求解器本身也需要并行化

➢ 求解NS方程所需要的线性方程组求解器有：

Linear Solver	Target
Conjugate Gradient (CG)	压强泊松方程
Stabilized Bi-Conjugate Gradient (BiCGstab)	动量方程

➢ 以共轭梯度法为例，每次迭代需要两次全局通信

Compute $r_0 = b - Ax_0$, $z_0 = M^{-1}r_0$, $p_0 = z_0$

For $j = 0, 1, \cdots,$ until convergence, Do:

$\boxed{\alpha_j = (r_j, z_j)/\boxed{(Ap_j, p_j)}}$

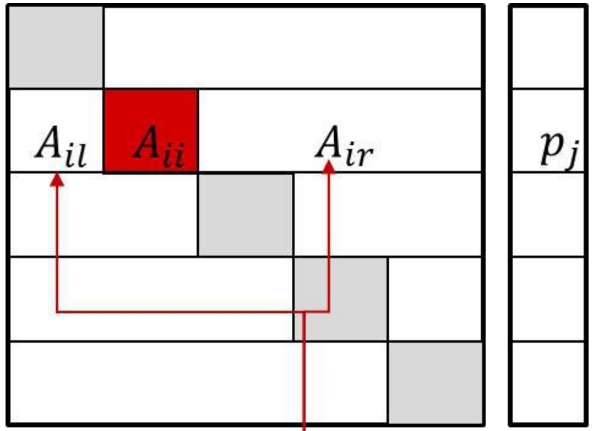
$x_{j+1} = x_j + \alpha_j p_j$

$r_{j+1} = r_j - \alpha_j Ap_j$

$z_{j+1} = M^{-1} r_{r+1}$

$\boxed{\beta_j = (r_{j+1}, z_{j+1})/(r_j, z_j)}$

$p_{j+1} = z_{j+1} + \beta_j p_j$



Contribution of ghost cells

To synchronize α_j and β_j across processors:
- MPI-collective and broadcasting operations required on numerator and denominators

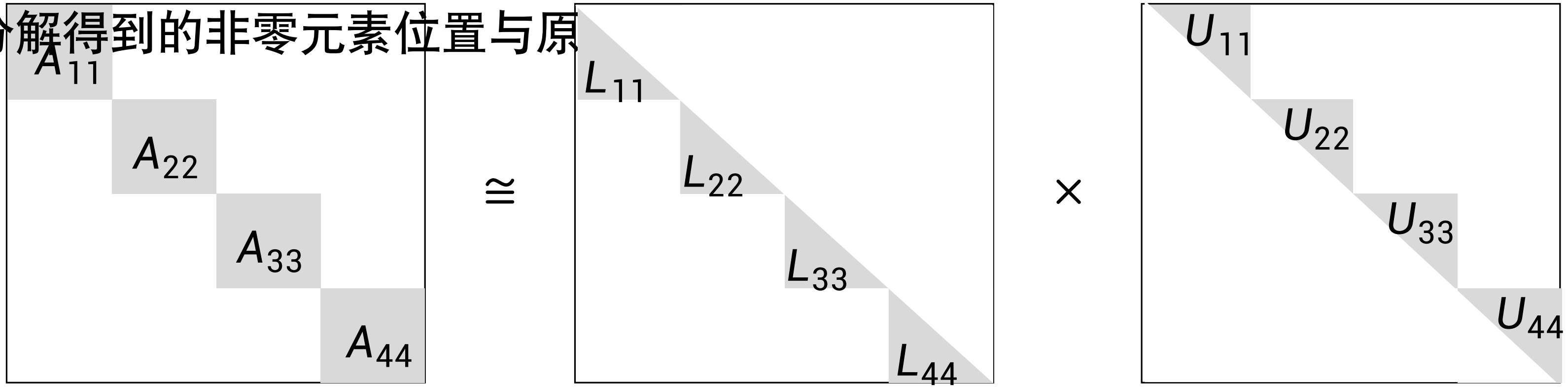
针对大规模并行计算，线性方程组预处理也需要进行修改

➢ 常见的预处理方法（为什么要预处理？线性方程组的条件数、收敛性）：

Preconditioner	**Source**	**Target**
Incomplete Cholesky (IC)	Implemented	PCG
Incomplete LU (ILU)	Calling MKL	GMRES, PBiCG

➢ 以矩阵的ILU分解为例，有以下难点：

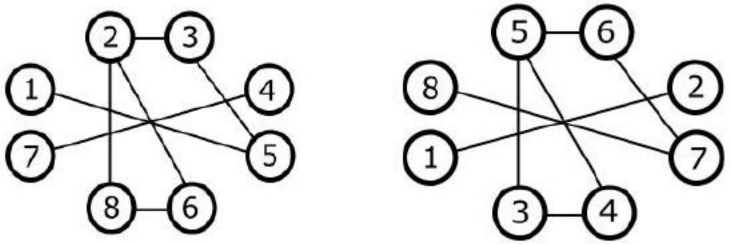
- ➢ 每个处理器存储本地分块，LU依次分解效率很低 —— 只对A_{ii}做局部LU分解、忽略非对角块的元素
- ➢ ILU分解得到的非零元素位置与原 —— (ILU0，忽略新增的非零元素



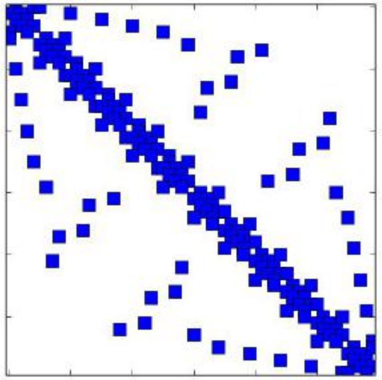
线性方程组的大规模并行求解

为了改进线性方程组的收敛性，可以对网格进行重编号

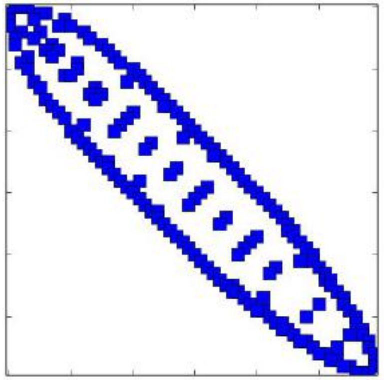
- 重编号可以显著降低矩阵的带宽（bandwidth）



Original cell numbering & connectivity　　RCM updated cell numbers

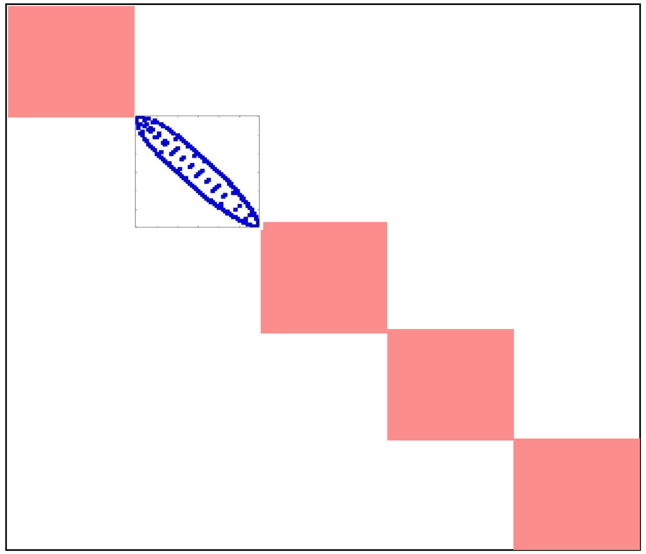


原始非零元素位置



排序后的非零元素位置

- 在什么时候进行网格重编号?
  - 选择一：先重编号然后进行并行分块——注意分块后的重编号问题!
  - 选择二：先完成并行分块，然后局部进行重编号



局部重编号示意图

PART 04

计算流体力学实践

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

面向对象的程序架构：简单直接的偏微分方程和物理场的定义方式

- 标量场、矢量场的简单模板化定义方式
- 计算流体力学的算子分为时间导数、对流项、扩散项、源项
- 基于SIMPLE算法，全局线性方程组的离散组装方法都可以模板化
- 举例：动量方程的简单离散

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p$$

```
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

场的定义方式

```
solve
(
      fvm::ddt(rho, U)
    + fvm::div(phi, U)
    - fvm::laplacian(mu, U)
    ==
    - fvc::grad(p)
)
```

动量方程定义

Term description	Implicit/explicit	Mathematical expression	fvm::/fvc:: functions
Laplacian	Implicit/Explicit	$\nabla \cdot \Gamma \nabla \phi$	laplacian(Gamma,phi)
Time derivative	Implicit/Explicit	$\partial \phi / \partial t$	ddt(phi)
		$\partial \rho \phi / \partial t$	ddt(rho, phi)
Convection	Implicit/Explicit	$\nabla \cdot (\psi)$	div(psi, scheme)
		$\nabla \cdot (\psi \phi)$	div(psi, phi, word)
			div(psi, phi)
Source	Implicit	$\rho \phi$	Sp(rho, phi)
	Implicit/Explicit		SuSp(rho, phi)

算子化的方程各项定义

Open Field Operation And Manipulation

- **下载算例**

  git clone git@gitlab.com:sjtu-saa-cfd/engineeringclass.git .

  git clone https://gitlab.com/sjtu-saa-cfd/engineeringclass.git .

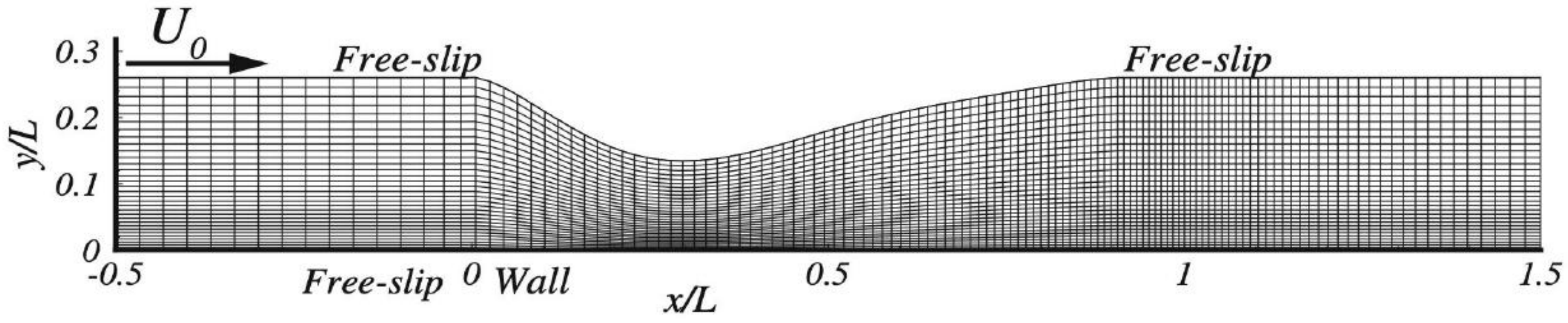
  Wget https://gitlab.com/sjtu-saa-cfd/engineeringclass/-/archive/main/engineeringclass-main.tar.gz

- **加载OpenFOAM**

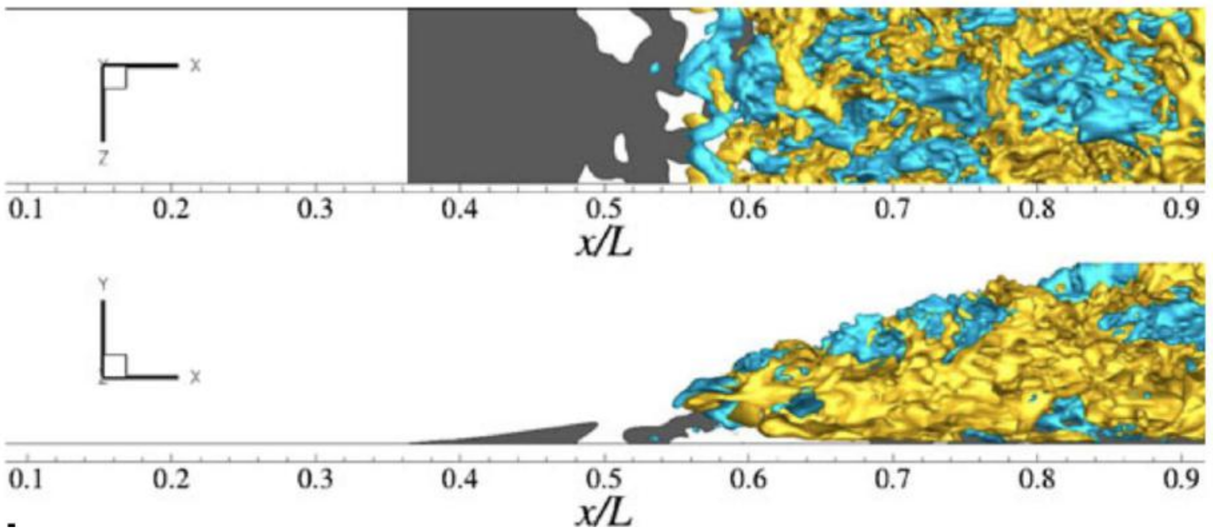
armlogin.hpc.sjtu.edu.cn	module load openfoam/v2012-gcc-10.3.1
sylogin.hpc.sjtu.edu.cn	module load openfoam/2206-intel-2021.4.0
pilogin.hpc.sjtu.edu.cn	module load openfoam/1912-gcc-7.4.0-openmpi

压强梯度引起的边界层流动分离（Lardeau et al., 2012）

➢ 计算网格



➢ 计算结果



Lardeau, Sylvain, Michael Leschziner, and Tamer Zaki. "Large eddy simulation of transitional separated flow over a flat plate and a compressor blade." *Flow, turbulence and combustion* 88 (2012): 19-44.

SHANGHAI JIAO TONG UNIVERSITY

上海交通大學

THANKS