icoFoam解析

李东岳

1. 引言

计算流体力学 (CFD) 采用计算机模拟流体的流动行为。流体动力学包含各种类型，如可压缩流动、不可压缩流动、多相流动以及化学反应、组分输运等。所有的这些流动性为都可以用偏微分方程来描述。例如，不可压缩牛顿流体的控制方程可以表示为 (Ferziger and Peric, 2012)：

$$\nabla\cdot\mathbf{U}=0, \tag{1}$$

$$\frac{\partial\mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{U}\mathbf{U})=-\nabla\frac{p}{\rho}+\nabla\cdot(\nu\nabla\mathbf{U}), \tag{2}$$

其中ρ为密度，$\mathbf{U}$为速度，p为压力，ν为粘度。该方程具有以下特点：（1）方程(2)中左边第二项是关于$\mathbf{U}$乘积的偏导数，这种未知量和未知量乘积的问题构成非线性问题，CFD对非线性问题需要特殊处理。另一方面，非线性的双曲问题的解可能会存在间断（如激波）。激波通常存在于高超声速的欧拉问题求解中。同时，非线性项也是湍流在数学方程中的体现；（2）方程(2)的数学特征为抛物线。不同数学特征的问题需要调用不同的时间离散格式，隐性时间格式更有利于求解抛物线问题。若方程(2)中省略若干项则会改变方程的数学特征，例如若将方程右侧置为0，则变为双曲特征的欧拉方程。欧拉方程得益于其双曲特性，可采用迎风类显性算法推进，各种基于有限体积法的高分辨率格式因此而生（交错网格中心格式、中心-迎风格式等）；（3）方程(1)和(2)中存在未知量压力p，同时压力和速度是耦合在一起的（二者相互影响），但并没有单独的压力方程。这导致压力的求解需要特殊的策略。这也是CFD中压力基SIMPLE/PISO算法、密度基、耦合/解耦算法要处理的问题。另一方面，方程(2)右侧存在压力梯度项，所有类似的梯度项都需要特殊的数值处理否则会引起数值震荡；（4）方程(1)和(2)为宏观方程，调用了宏观假定，其起源为玻尔兹曼方程（又名动理学方程）。在更底层的介尺度研究领域，方程(1)和(2)也即从介尺度模型演化的宏观二阶矩模型。在无压力无粘性的条件下具备弱双曲特征。由于失去了高阶矩的统计学特征，因此方程(1)和(2)在某些情况下是不适用的；在考虑适当的简化情况下，方程(1)和(2)可被称之为Navier-Stokes (N-S) 方程。除了上述数学特征，求解N-S方程可分为不同的方法如有限体积法、有限差分法以及谱方法等。有限体积法得益于其天然守恒的数学特征，被大量的植入于CFD软件（如ANSYS Fluent、OpenFOAM等）中用于流体动力学计算模拟。

icoFoam为OpenFOAM中一个基于有限体积法的，用于求解不可压缩牛顿流体N-S方程的求解器。它可用于模拟层流流动或流场剪切力引致的湍流准直接模拟 (quasi-DNS)，也被称之为准DNS。在这里需要注意的是，得益于谱方法以及有限差分法高阶精度的易用性，DNS计算通常采用谱方法或有限差分法。有限体积法由于本身在计算通量时引入的中点插值法则，大大限制了有限体积法的精度。然而Komen等通过研究圆管内的湍流流动，认为在网格分辨率足够

的情况下，使用有限体积法同样可以获得高质量的平均速度以及脉动速度流场 (Komen et al., 2014) 。Komen等进一步的使用有限体积法进行准DNS计算，并与有限体积法大涡模拟进行标定，认为大涡模拟方法会引致一定的数值耗散 (Komen et al., 2017)。Axtmann与Rist也使用OpenFOAM进行了准DNS直接模拟，并研究了编译器与并行计算的特性 (Axtmann and Rist, 2016)。liu等通过icoFoam对开口顶盖驱动流进行了准直接模拟 (Liu et al., 2016)，进一步验证了使用OpenFOAM进行准直接模拟的可行性。

在icoFoam中，压力和速度的耦合通过瞬态PISO算法进行计算 (Issa, 1986) ，且没有考虑其他体积力（如重力等）。除去三个基本求解器 laplacianFoam，potentialFoam，scalarTransportFoam，icoFoam为一个最基本的求解器范例，可用于理解 OpenFOAM中压力速度耦合的框架策略。需要注意的是对于流场内温度变化较大，但可忽略浮力以及重力的可压缩求解器，可选用rhoPimpleFoam。附加重力的可压缩浮力驱动流求解器可选用buoyantPimpleFoam。本文从最基本的N-S方程在笛卡尔网格下的有限体积离散开始推导，介绍icoFoam求解器中的算法并和OpenFOAM植入的代码相对应，可用于理解N-S方程在有限体积法中的离散以及瞬态PISO算法在OpenFOAM中的实现。需要注意的是，由于为不可压缩流体求解器，下文中p的单位为压力除以密度的单位，即$\mathrm{m}^2\cdot\mathrm{s}^{-2}$。在下文中，$p$也对应方程(2)中的$p/\rho$。

2. 控制方程与算法

2.1 有限体积法离散

首先对方程(2)中的时间项进行对速度$\mathbf{U}$关于时间t的欧拉全隐离散有 (Jasak, 1996)：

$$\int\int \frac{\partial \mathbf{U}}{\partial t}\mathrm{d}V\mathrm{d}t = \left(\mathbf{U}_\mathrm{P}^{*} - \mathbf{U}_\mathrm{P}^{t}\right) V_\mathrm{P}, \tag{3}$$

对流项隐性离散：

$$\int\int \nabla\cdot(\mathbf{U}\mathbf{U})\mathrm{d}V\mathrm{d}t = \int\int \mathbf{U}\mathbf{U}\cdot\mathrm{d}\mathbf{S}\mathrm{d}t = \sum\left(\mathbf{U}^{*}\mathbf{U}^{t}\right)_f\cdot\mathbf{S}_f\Delta t = \Delta t\sum F_f^t\mathbf{U}_f^{*}, \tag{4}$$

拉普拉斯项隐性离散：

$$\int\int \nabla\cdot(\nu\nabla\mathbf{U})\mathrm{d}V\mathrm{d}t = \int\int \nu\nabla\mathbf{U}\cdot\mathrm{d}\mathbf{S}\mathrm{d}t = \sum\left(\nu\nabla\mathbf{U}^{*}\right)_f\cdot\mathbf{S}_f\Delta t = \Delta t\sum\nu(\nabla\mathbf{U}^{*})_f\cdot$$

压力项显性离散：

$$\int\int \nabla p\mathrm{d}V\mathrm{d}t = \int\int p\mathrm{d}\mathbf{S}\mathrm{d}t = \Delta t\sum\left(p_f^t\mathbf{S}_f\right), \tag{6}$$

其中上标^{t}表示为当前的时间步（已知），上标*表示预测步（待求），下标$_f$表示网格单元面上的值，V_P表示网格单元体积，$\mathbf{S}_f$表示网格单元的各个面的面矢量，Δt表示时间步长，F_f^t为通量，ν为动力粘度。方程(5)中$(\nabla\mathbf{U}^{*})_f\cdot\mathbf{S}_f$需要进一步处理，其中$\nabla\mathbf{U}^{*}$为定义在体心的量，即网格体心的速度梯度。$(\nabla\mathbf{U}^{*})_f$为定义在面心的量，即网格面心的速度梯度。为使用紧致基架点防止数值震荡，$(\nabla\mathbf{U}^{*})_f\cdot\mathbf{S}_f$通常表示为面法向梯度与面矢量的模的乘积：

$$(\nabla \mathbf{U}^*)_f \cdot \mathbf{S}_f = \left((\nabla \mathbf{U}^*)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} \right) \cdot |\mathbf{S}_f| \tag{7}$$

其中$(\nabla \mathbf{U}^*)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|}$表示面法向梯度，有

$$(\nabla \mathbf{U}^*)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} = \frac{\mathbf{U}_\mathrm{N}^* - \mathbf{U}_\mathrm{P}^*}{|\mathbf{d}|}, \tag{8}$$

其中$\mathbf{d}$表示网格单元体心之间矢量差（距离矢量），下标$_\mathrm{N}$表示相邻网格单元的速度，下标$_\mathrm{P}$表示当前网格单元的速度。速度的面法向梯度在三维情况下为一个矢量（变量的面法向梯度与变量的阶数相同）。在这里需要注意的是，方程(8)只对矩形网格精准，网格非正交性增加，需要特定的数值处理提高其计算精准性。将方程(3)-(7)代入到方程(2)有：

$$\frac{\mathbf{U}_\mathrm{P}^* - \mathbf{U}_\mathrm{P}^t}{\Delta t} V_\mathrm{P} + \sum F_f^t \mathbf{U}_f^* = \sum \nu \nabla_f \mathbf{U}^* |\mathbf{S}_f| - \sum p_f^t \mathbf{S}_f. \tag{9}$$

需要注意的是，方程(2)中的左边第二项（对流项）是非线性的。在求解的过程中，要么选择非线性求解器，要么将对流项线性化。由于非线性求解器非常复杂，因此在OpenFOAM均采用线性化处理。具体的，在对方程(9)中的左边第二项对流项离散中，其中的通量F_f^t采用当前已知时间步的速度$\mathbf{U}^t$来计算，同时保留$\mathbf{U}_\mathrm{P}^*$为未知量。这种将高阶非线性项降为一阶线性项的过程即为线性化操作。线性化的一个问题即为通量速度的信息有所滞后 (Jasak, 1996)。

2.2. 动量预测

方程(9)中$\mathbf{U}_f^*$被定义为面上的预测速度。为方便与OpenFOAM代码对应，在时间步内第一次通过求解动量方程获得的速度叫预测速度。面速度需要从体心速度进行插值来获得，在此步可以引入各种插值格式。假设在均一网格上使用中心线性格式：

$$\mathbf{U}_f^* = \frac{\mathbf{U}_\mathrm{P}^* + \mathbf{U}_\mathrm{N}^*}{2}. \tag{10}$$

$$p_f^t = \frac{p_\mathrm{P}^t + p_\mathrm{N}^t}{2}. \tag{11}$$

将方程(7)，(10)，(11)代入到方程(9)有

$$\frac{\mathbf{U}_\mathrm{P}^* - \mathbf{U}_\mathrm{P}^t}{\Delta t} V_\mathrm{P} + \sum F_f^n \frac{\mathbf{U}_\mathrm{P}^* + \mathbf{U}_\mathrm{N}^*}{2} = \sum \nu |\mathbf{S}_f| \frac{\mathbf{U}_\mathrm{N}^* - \mathbf{U}_\mathrm{P}^*}{|\mathbf{d}|} - \sum \frac{p_\mathrm{P}^t + p_\mathrm{N}^t}{2} \mathbf{S}_f, \tag{12}$$

$$\left(\frac{1}{\Delta t} + \frac{1}{V_\mathrm{P}} \sum \frac{F_f^t}{2} + \frac{1}{V_\mathrm{P}} \sum \nu \frac{|\mathbf{S}_f|}{|\mathbf{d}|} \right) \mathbf{U}_\mathrm{P}^* =$$

$$- \sum \frac{1}{V_\mathrm{P}} \left(\frac{F_f^t}{2} - \nu \frac{|\mathbf{S}_f|}{|\mathbf{d}|} \right) \mathbf{U}_\mathrm{N}^* + \frac{1}{\Delta t} \mathbf{U}_\mathrm{P}^t - \frac{1}{V_\mathrm{P}}$$

将上式简写为

$$A_{\mathrm{P}}\mathbf{U}^*_{\mathrm{P}}+\sum A_{\mathrm{N}}\mathbf{U}^*_{\mathrm{N}}=S^t_{\mathrm{P}}-\frac{1}{V_{\mathrm{P}}}\sum\frac{p^t_{\mathrm{P}}+p^t_{\mathrm{N}}}{2}\mathbf{S}_f, \tag{14}$$

其中

$$A_{\mathrm{P}}=\frac{1}{\Delta t}+\frac{1}{V_{\mathrm{P}}}\sum\frac{F^t_f}{2}+\frac{1}{V_{\mathrm{P}}}\sum\nu\frac{|\mathbf{S}_f|}{|\mathbf{d}|}, \tag{15}$$

$$A_{\mathrm{N}}=\frac{1}{V_{\mathrm{P}}}\left(\frac{F^t_f}{2}-\nu\frac{|\mathbf{S}_f|}{|\mathbf{d}|}\right), \tag{16}$$

$$S^t_{\mathrm{P}}=\frac{1}{\Delta t}\mathbf{U}^t_{\mathrm{P}}. \tag{17}$$

可以看出在某个时间步内A_{P}、A_{N}、和S^t_{P}保持不变（注意A_{N}、$\sum A_{\mathrm{N}}\mathbf{U}_{\mathrm{N}}$、$-\sum A_{\mathrm{N}}\mathbf{U}_{\mathrm{N}}+S_{\mathrm{P}}$的区别）。求解方程(14)即可获得预测速度$\mathbf{U}^*_{\mathrm{P}}$。方程(14)即为OpenFOAM中的动量预测方程。需要注意的是，动量预测步骤并不是必须的，这主要和速度压力耦合求解策略有关。如果不调用动量预测步骤，则$\mathbf{U}^*_{\mathrm{P}}=\mathbf{U}^t_{\mathrm{P}}$。

2.3. 压力泊松方程

N-S方程求解的关键问题之一在于并没有压力的方程出现。仔细分析得知上文中求解动量方程获得了速度，但连续性方程并不是关于压力的方程。压力泊松方程的构建正式基于连续性方程而来，其通过对动量方程继续做散度并相加获得。考虑最终收敛的情况下，对连续性方程$\nabla\cdot\mathbf{U}=0$进行离散后的形式为

$$\sum\left(\mathbf{U}^{t+\Delta t}_{\mathrm{P},f}\cdot\mathbf{S}_f\right)=0. \tag{18}$$

如果能用压力表示方程(18)中的$\mathbf{U}^{t+\Delta t}_{\mathrm{P},f}$，是不是就可以得出压力泊松方程了呢？答案是肯定的。首先，方程(14)可以写为：

$$A_{\mathrm{P}}\mathbf{U}^*_{\mathrm{P}}+\sum A_{\mathrm{N}}\mathbf{U}^*_{\mathrm{N}}=S^t_{\mathrm{P}}-\frac{1}{V_{\mathrm{P}}}\sum p^t_f\mathbf{S}_f, \tag{19}$$

收敛情况下为：

$$A_{\mathrm{P}}\mathbf{U}^{t+\Delta t}_{\mathrm{P}}+\sum A_{\mathrm{N}}\mathbf{U}^{t+\Delta t}_{\mathrm{N}}=S^t_{\mathrm{P}}-\frac{1}{V_{\mathrm{P}}}\sum p^{t+\Delta t}_f\mathbf{S}_f, \tag{20}$$

若将方程(20)减去(19)有：

$$A_{\mathrm{P}}\mathbf{U}'_{\mathrm{P}}+\sum A_{\mathrm{N}}\mathbf{U}'_{\mathrm{N}}=-\frac{1}{V_{\mathrm{P}}}\sum p'_f\mathbf{S}_f, \tag{21}$$

其中$\mathbf{U}'_{\mathrm{P}}=\mathbf{U}^{n+1}_{\mathrm{P}}-\mathbf{U}^r_{\mathrm{P}}$（其他同理）。在这里未调用任何略去临点的假定，方程(20)是严格成立的（精准的）。对方程(20)移项有

$$\mathbf{U}^{t+\Delta t}_{\mathrm{P}}=\mathbf{HbyA}^{t+\Delta t}_{\mathrm{P}}-\frac{1}{A_{\mathrm{P}}}\frac{1}{V_{\mathrm{P}}}\sum p^{t+\Delta t}_f\mathbf{S}_f. \tag{22}$$

其中有$\mathbf{HbyA}$

$$\mathbf{HbyA}_{\mathrm{P}}^{t+\Delta t}=\frac{1}{A_{\mathrm{P}}}\Big(-\sum A_{\mathrm{N}}\mathbf{U}_{\mathrm{N}}^{t+\Delta t}+S_{\mathrm{P}}^{n}\Big). \tag{23}$$

类似的，面上的速度可以通过下面的公式获得：

$$\mathbf{U}_{\mathrm{P},f}^{t+\Delta t}=\mathbf{HbyA}_{f}^{t+\Delta t}-\frac{1}{A_{\mathrm{P},f}}\left(\frac{1}{V_{\mathrm{P}}}\sum p_{f}^{t+\Delta t}\mathbf{S}_{f}\right)_{f}. \tag{24}$$

将方程(24)代入到方程(18)有：

$$\sum\left(\mathbf{HbyA}_{f}^{t+\Delta t}-\frac{1}{A_{\mathrm{P},f}}\left(\frac{1}{V_{\mathrm{P}}}\sum p_{f}^{t+\Delta t}\mathbf{S}_{f}\right)_{f}\right)\cdot\mathbf{S}_{f}=0, \tag{25}$$

$$\sum\mathbf{HbyA}_{f}^{t+\Delta t}\cdot\mathbf{S}_{f}=\sum\frac{1}{A_{\mathrm{P},f}}\left(\frac{1}{V_{\mathrm{P}}}\sum p_{f}^{t+\Delta t}\mathbf{S}_{f}\right)_{f}\cdot\mathbf{S}_{f}, \tag{26}$$

方程(26)即下述方程的离散形式：

$$\nabla\cdot\mathbf{HbyA}^{t+\Delta t}=\nabla\cdot\left(\frac{1}{A}\nabla p^{t+\Delta t}\right). \tag{27}$$

方程(27)即为压力泊松方程。若有$\mathbf{HbyA}^{t+\Delta t}$，则可求得收敛的压力。需要注意的是，$\nabla\cdot\left(\frac{1}{A}\nabla p^{t+\Delta t}\right)$可以通过先求梯度再求散度进行，也可以直接通过拉普拉斯积分进行。方程(26)右端即先求梯度再求散度的方法，这种方法会引起数值震荡。若通过拉普拉斯并进行高斯积分进行离散，则会调用紧致格式防止数值震荡 (Li, 2019)。

2.4. 迭代算法

可以看出，最后求解的速度$\mathbf{U}_{\mathrm{P}}^{t+\Delta t}$和压力$p^{t+\Delta t}$应该符合方程(14)和(27)，分别对应动量方程和连续性方程。然而目前，我们只有通过2.2节动量预测步骤求出来的$\mathbf{U}_{\mathrm{P}}^{*}$。$\mathbf{HbyA}_{\mathrm{P}}^{t+\Delta t}$和$p^{t+\Delta t}$均为未知的。压力泊松方程不能求解。1986年Issa提出了PISO算法可以解决这个问题 (Issa, 1986)。PISO算法在提出时主要针对不可压缩非稳态计算，其为一种在时间步内非迭代的算法，在随后也被拓展用于稳态问题以及可压问题中。PISO算法通过对当前时间步内的多次修正来获得最终的$\mathbf{U}_{\mathrm{P}}^{t+\Delta t}$和$p^{t+\Delta t}$。依据PISO算法，在方程(21)的基础上引入略去临点影响的假定有：

$$A_{\mathrm{P}}\mathbf{U}_{\mathrm{P}}'=-\frac{1}{V_{\mathrm{P}}}\sum p_{f}'\mathbf{S}_{f}, \tag{28}$$

将方程(28)和(19)相加有

$$A_{\mathrm{P}}\mathbf{U}_{\mathrm{P}}^{**}+\sum A_{\mathrm{N}}\mathbf{U}_{\mathrm{N}}^{*}=S_{\mathrm{P}}^{n}-\frac{1}{V_{\mathrm{P}}}\sum p_{f}^{*}\mathbf{S}_{f}, \tag{29}$$

其中$\mathbf{U}_{\mathrm{P}}^{**}=\mathbf{U}_{\mathrm{P}}^{*}+\mathbf{U}_{\mathrm{P}}'$，$p_{f}^{*}=p_{f}^{t}+p_{f}'$。相比方程(20)，方程(29)由于忽略了临点的影响并不是精准的。对方程(29)进行移项有

$$\mathbf{U}_{\mathrm{P}}^{**}=\mathbf{HbyA}_{\mathrm{P}}^{*}-\frac{1}{A_{\mathrm{P}}}\frac{1}{V_{\mathrm{P}}}\sum p_{f}^{*}\mathbf{S}_{f}. \tag{30}$$

$$\mathbf{HbyA}_{\mathrm{P}}^{*}=\frac{1}{A_{\mathrm{P}}}\left(-\sum A_{\mathrm{N}}\mathbf{U}_{\mathrm{N}}^{*}+S_{\mathrm{P}}^{n}\right). \tag{31}$$

$$\mathbf{U}_{\mathrm{P},f}^{**}=\mathbf{HbyA}_{f}^{*}-\frac{1}{A_{\mathrm{P},f}}\left(\frac{1}{V_{\mathrm{P}}}\sum p_{f}^{*}\mathbf{S}_{f}\right)_{f}. \tag{32}$$

方程(32)中可参考方程(25)-(27)的步骤组建压力泊松方程，即

$$\nabla\cdot\left(\frac{1}{A}\nabla p^{*}\right)=\nabla\cdot\mathbf{HbyA}^{*} \tag{33}$$

对方程(33)求解后有p^*。将p^*回代到方程(32)的时候，有$\mathbf{U}_{\mathrm{P}}^{**}$。但需要注意的是，方程(32)并不是精准的。因此$p^*$和$\mathbf{U}_{\mathrm{P}}^{**}$并不是最终时间步$t+\Delta t$的结果。通过多次更新速度，并求解压力泊松方程，这种滞后性逐渐消除并可最终忽略。总而言之，PISO算法中的的迭代过程可以表示为下面几个步骤：

1. 依据初始条件求解预测速度$\mathbf{U}^*$；
2. 通过速度组建$\mathbf{HbyA}^*$；
3. 求解方程(33)求解压力p^*；
4. 通过方程(32)依据压力p^*以及$\mathbf{HbyA}^*$更新速度有$\mathbf{U}^{**}$；
5. 回到第二步迭代求解几次即可；

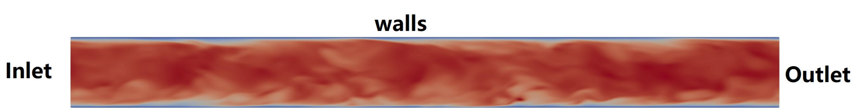
3. 验证算例

3.1. Quasi-DNS直接模拟

icoFoam自带若干算例。在OpenFOAM-6中，自带的算例主要为以下几个：

- `cavity`：顶盖驱动流问题；
- `elbow`：非结构网格弯头流动问题；

上述算例在《OpenFOAM用户指南》已经进行了非常详细的阐述。在此不做过多介绍。除上述算例外，icoFoam也可以进行直接模拟。icoFoam不附加任何湍流模型直接求解N-S方程，并采用二阶有限体积法进行离散求解。因此可用作为一个直接模拟求解器进行直接模拟。一些文章中更严谨的将其称为准直接模拟:quasi-DNS。另外，在这里强调OpenFOAM中的icoFoam、pisoFoam以及pimpleFoam在不使用湍流模型的情况下是相同的。本算例对标相关文献的研究工作 (Komen et al., 2017)。



算例网格为采用blockMesh生成的3D纯六面体结构网格。左侧为周期速度进口，右侧为周期速度出口，上下为用于附加剪切的壁面。需要注意的是，对于DNS以及LES，流场本身的湍流结构以及进口边界条件非常敏感。在不合理的初始条件下，二阶精度的有限体积法可能并不会捕获精细的涡结构。因此初始场应该尽可能的精准。本算例初始场通过boxTurb给定湍流涡结构之后，运行了200个周期，作为稳定的初始场。$\mathrm{Re}_\tau \approx 180$。

- 初始场 `p`： 内部即均一分布 `uniform 0`；
- 初始场速度 `U`： 内部速度为0.1335，即均一分布 `uniform (0.1335 0 0)`；
- 边界场 `p`： 边界全部为 `cyclic`， 壁面为 `zeroGradient` 零法向梯度；
- 边界场 `U`： 边界全部为 `cyclic`， 壁面为 `noSlip` 无滑移边界；

constant文件夹的物性主要设置如下：

- 粘度模型选择层流，即 `laminar`；
- 流体的动力粘度为2e-5；
- fvOptions文件用于指定压力梯度，其中meanVelocityForce用于指定fvOptions的类型。UBar平均速度指定 `(0.1335 0 0)`；

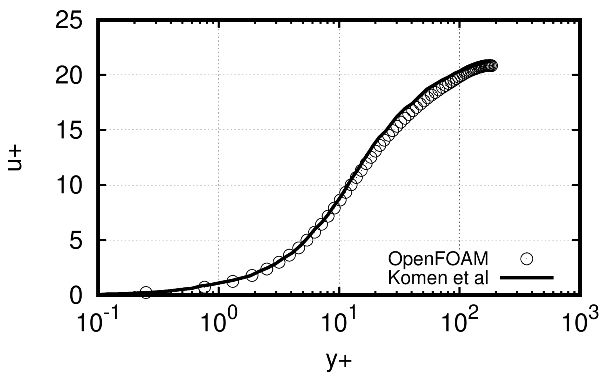
相关算例运行首先需要执行网格转换命令：

```
blockMesh
```

网格生成完毕，后直接运行直接模拟求解器运行即可：

```
pisoFoam
```

在这里提醒，pisoFoam是icoFoam的进阶版本。由于icoFoam中不支持fvOptions，因此本算例采用pisoFoam进行计算。本算例比较耗费计算机资源，采用12核并行计算大约需要10个小时。在运行之后，针对平均速度 `UMean` 即可以求出u+以及y+（如下图）。



点击下载DNS算例。因为初始场信息已经经过计算200个周期，因此文件较大，约65m。

3.2. 圆柱绕流

下面，采用icoFoam对圆柱绕流进行计算模拟。相关算例由CFD中文网用户波流力提供。算例采用 `blockMesh` 生成的非正交四面体网格。左侧为进口，右侧为出口，计算域内存在一个圆柱。上下边界为对称面。0文件夹下的相关设置如下：

- 初始场 `p` ： 内部即均一分布 `uniform 0` ；
- 初始场速度 `U` ： 内部速度为1，即均一分布 `uniform (1 0 0)` ；
- 边界场 `p` ： 出口为 `fixedValue` 固定值边界条件，进口为 `zeroGradient` 零法向梯度；上下采用 `symmetryPlane` 对称边界条件；
- 边界场 `U` ： 进口为 `fixedValue` 固定值边界条件，出口为 `zeroGradient` 零法向梯度；上下采用 `symmetryPlane` 对称边界条件；

constant文件夹的物性主要设置如下：

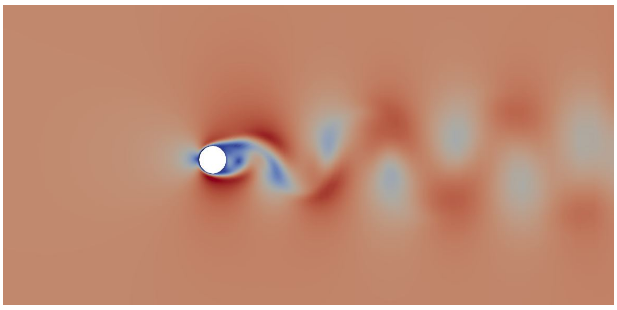
- 粘度模型选择层流，即 `laminar` ；
- 流体的动力粘度为0.01，一个非常粘的流体，雷诺数为200；

相关算例运行首先需要执行网格转换命令：

```
blockMesh
```

网格生成完毕，后直接运行直接模拟求解器运行即可：

```
icoFoam
```



点击下载圆柱绕流算例

更新历史
2020.07.19warner bestucan | 2020.07.07周佳其 | 2020.03.04周佳其 | 2019.10.12Yongbo | 2019.6.12chaoscfd | 2018.12.08丁长友 | 2018.11.07ghoulyy | 2018.08.22于鸿翔 | 2018.07.05尹胜强 | 2018.05.21JimShi | 2018.05.08张某人