scalarTransportFoam解析

李东岳

1. 引言

scalarTransportFoam是OpenFOAM3个最基本的求解器之一，用于求解标量（passive scalar）传输问题。其为一个稳态或瞬态标量传输求解器。在这个求解器中，用户需要在constant/transportProperties当中设置标量的扩散率DT。依据标量的定义，其对流场不具有影响作用，即为单向耦合，因此我们没有必要实时的计算流场。因此在0文件夹下提供一个不随时间变化的速度场U即可。

scalarTransportFoam中植入的方程为标量传输方程：

$$\frac{\partial T}{\partial t}+\nabla\cdot(\mathbf{U}T)-\nabla\cdot(\nabla D_{\mathrm{T}}T)=S \tag{1}$$

其中T表示被传输的标量，$\mathbf{U}$表示传输速度，D_{T}表示扩散系数。

2. 代码分析

在求解方程(1)的时候，需要定义若干个场。进入createFields.H文件：（大部分代码略，请参考laplacianFoam解析）

```
Info<< "Reading/calculating face flux field phi\n" << endl;

surfaceScalarField phi
(
    IOobject
    (
        "phi",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT,
        IOobject::AUTO_WRITE
    ),
    linearInterpolate(U) & mesh.Sf()//公式(1)
);
```

在上面的代码中，phi是一个在OpenFOAM中非常常见的定义，即为通量（请参考：CFD中的通量）。其在此处被定义为：

$$\mathbf{phi}=\mathbf{U}_f\cdot\mathbf{S}_f \tag{2}$$

然后进入scalarTransportFoam.C：

```
#include "fvCFD.H"
//在OpenFOAM的所有求解器中都可以看到这个头文件，它涉及到构建时间、组建矩阵、有限体积离散、组
//建议在自定义的求解器中必备此项。

#include "fvOptions.H"
//通过fvOption来修正源项，主要用于在运行时添加多孔介质、MRF多重参考系、隐形显性热源等，如果

#include "simpleControl.H"
//包含SIMPLE循环头文件，使用SIMPLE循环必须进行包含。

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"//设置算例的根目录，必备头文件。请忽略
    #include "createTime.H"//创建时间对象，大部分求解器都需要此文件。请忽略
    #include "createMesh.H"//创建网格对象，必备头文件。请忽略

    simpleControl simple(mesh);//对于采用SIMPLE算法的算例，必备此项，请忽略

    #include "createFields.H"//包含上文分析过的createFields.H头文件
    #include "createFvOptions.H"//创建源项，无需源项可删除。请忽略

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nCalculating scalar transport\n" << endl;

    #include "CourantNo.H" //计算库郎数，见下文分析

    while (simple.loop()) //开始SIMPLE循环，采用SIMPLE算法必备语句
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        while (simple.correctNonOrthogonal())
        {
            fvScalarMatrix TEqn
            (
                fvm::ddt(T) //时间项，公式(1)左边第一项
              + fvm::div(phi, T) //对流项，公式(1)左边第二项
              - fvm::laplacian(DT, T) //扩散项，公式(1)左边第三项
             ==
                fvOptions(T) //源项，公式(1)右边
            ); //组建TEqn，公式(1)

            TEqn.relax(); //对上述方程松弛
            fvOptions.constrain(TEqn); //对方程系数进行源项限定
            TEqn.solve(); //对上述方程求解
            fvOptions.correct(T); //对T进行修正
        }

        runTime.write();
    }

    Info<< "End\n" << endl;

    return 0;
}
```

最后我们分析Courant.H。CFD计算中通常要求Courant数小于1，在某些多相流情况下小于0.5是最好的。然而某些特定的数值格式可以使用较大的Courant数。Courant数用来判断是否满足CFL稳定性标准的无量纲数，在一维的情况下定义为

$$\mathrm{Co} = \frac{|u|\Delta t}{\Delta x} \tag{3}$$

其中u为网格单元中心x方向的速度，Δx表示网格单元的x方向长度。在三维的情况下定义为

$$\mathrm{Co} = 0.5\frac{\Delta t \sum_f |\phi_f|}{\Delta V} \tag{4}$$

其中ϕ_f表示网格单元面f的通量，ΔV表示网格单元体积。方程(4)也被称之为面库朗数。由于通量守恒，进入网格单元的通量等于流出网格单元的通量，因此在计算面库朗数的时候，要乘以0.5。

```
scalar CoNum = 0.0;
//定义一个scalar（类似C++中的double），CoNum=0.0

scalar meanCoNum = 0.0;//同上

if (mesh.nInternalFaces()) //如果是网格的内部面，不考虑边界面
{
    scalarField sumPhi
    (
        fvc::surfaceSum(mag(phi))().internalField()
    );

    CoNum = 0.5*gMax(sumPhi/mesh.V().field())*runTime.deltaTValue();
    //公式(4)

    meanCoNum =
        0.5*(gSum(sumPhi)/gSum(mesh.V().field()))*runTime.deltaTValue();
}

Info<< "Courant Number mean: " << meanCoNum
    << " max: " << CoNum << endl;
```

更新历史

2018.03.31：重新排版，统一公式排版