Школа-семинар

«Расширенные возможности пакета OpenFOAM»

САМОСТОЯТЕЛЬНАЯ РАБОТА

М.В. Крапошин (НИЦ Курчатовский Институт)
О.И. Самоваров (Институт Системного Программирования РАН)
С.В. Стрижак (ГОУ ВПО МГТУ им. Баумана)

СОДЕРЖАНИЕ

- Гиперболические уравния

$$\frac{\partial \psi}{\partial t} + c\frac{\partial \psi}{\partial x} = 0 \qquad \frac{\partial^2 \psi}{\partial t^2} - c_0 \frac{\partial^2 \psi}{\partial x^2} = 0$$

- Параболические уравнения

$$\frac{\partial u}{\partial t} - c\frac{\partial^2 u}{\partial x^2} = 0$$

- Эллиптические уравнения

$$\frac{\partial^2 u}{\partial x^2} = 0$$

РЕШЕНИЕ УРАВНЕНИЙ ГИПЕРБОЛИЧЕСКОГО ТИПА (1)

- Дискретизация уравнений в OpenFOAM

$$\frac{\partial \psi}{\partial t} + c\frac{\partial \psi}{\partial x} = 0$$

$$\frac{\partial^2 \psi}{\partial t^2} - c_0\frac{\partial^2 \psi}{\partial x^2} = 0$$

hyper1Foam

```
solve
(
    fvm::ddt(psi)
    +
    fvm::div(phi,psi)
);
```

hyper1MulesFoam

```
MULES::explicitSolve(psi, phi,
phiPsi, 1, 0);
```

hyper2Foam

```
solve
(
    fvm::d2dt2(psi)
    -
    fvm::laplacian(c0,psi)
);
```

РЕШЕНИЕ УРАВНЕНИЙ ГИПЕРБОЛИЧЕСКОГО ТИПА (2)

- Инициализация полей hyper1Foam

```
Info<< "Reading field U\n" <<
endl;

volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

#   include "createPhi.H"
```

```
    Info<< "Reading field psi\n"
<< endl;

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

РЕШЕНИЕ УРАВНЕНИЙ ГИПЕРБОЛИЧЕСКОГО ТИПА (3)

- Инициализация полей hyper2Foam

```
IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ,
        IOobject::NO_WRITE
    )
);

Info<< "Reading diffusivity D\n" <<
endl;

dimensionedScalar c0
(
    transportProperties.lookup("c0")
);
```

```
Info<< "Reading field psi\n"
<< endl;

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

РЕШЕННИЕ УРАВНЕНИЙ ПАРАБОЛИЧЕСКОГО ТИПА

- Дискретизация уравнений в OpenFOAM

$$\frac{\partial T}{\partial t} - \nabla \cdot (\lambda \nabla T) = 0$$

```
parabFoam
for (int nonOrth=0; nonOrth<=simple.nNonOrthCorr(); nonOrth++)
{
    solve
    (
        fvm::ddt(T)
      - fvm::laplacian(DT, T)
    );
}
```

РЕШЕННИЕ УРАВНЕНИЙ ПАРАБОЛИЧЕСКОГО ТИПА

- Инициализация полей parabFoam

```
IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
         runTime.constant(),
         mesh,
         IOobject::MUST_READ,
         IOobject::NO_WRITE
    )
);
    Info<< "Reading diffusivity D\n"
<< endl;

dimensionedScalar DT
(
    transportProperties.lookup("DT")
);
```

```
Info<< "Reading field T\n" <<
endl;

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

РЕШЕНИЕ УРАВНЕНИЙ ЭЛЛИПТИЧЕСКОГО ВИДА (1)

- Дискретизация уравнений в OpenFOAM

$$\nabla^2 \phi = 0 = \nabla^2 p$$

```
fvScalarMatrix pEqn
(
    fvm::laplacian
    (
        dimensionedScalar
        (
            "1",
            dimTime/p.dimensions()*dimensionSet(0, 2, -2, 0, 0),
            1
        ),
        p
    )
 ==
    fvc::div(phi)
);
```

РЕШЕНИЕ УРАВНЕНИЙ ЭЛЛИПТИЧЕСКОГО ВИДА (2)

- Инициализация полей elliFoam

```
    Info<< "Reading field p\n" <<
endl;
    volScalarField p
    (
        IOobject
        (
            "p",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        mesh
    );

    p = dimensionedScalar("zero",
p.dimensions(), 0.0);
```

```
    Info<< "Reading field U\n" <<
endl;
    volVectorField U
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    U = dimensionedVector("0",
U.dimensions(), vector::zero);
```

СПАСБО ЗА ВНИМАНИЕ!

$$pV = \nu R T$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{U} = 0$$

fvm::ddt(rho) + fvc::div(phi)=0

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) - \nabla \cdot \left(\mu \frac{1}{2} \left(\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T \right) \right) = -\nabla p$$

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