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)

№1

СОДЕРЖАНИЕ

A) Разработка нового граничного условия 1-го типа

B) Разработка нового граничного условия 2-го типа

C) Разработка нового граничного условия смешанного типа

D) Разработка новой численной схемы

E) Разработка утилиты для препроцессинга

F) Разработка утилиты для постпроцессинга

G) Добавление нового уравнения в решатель

СОЗДАНИЕ БИБЛИОТЕКИ

- Создать папку myLib
- Создать в папке myLib/Make
- Изменить файлы Make/files, Make/options
- Создать подпапки с исходным кодом
- Изменить параметры сборки (Make/files, Make/options)
- Сборка — `wmake libso`

НАСТРОЙКИ СБОРКИ БИБЛИОТЕКИ

Make/files

```
conductiveHeatFlux/conductiveHeatFluxFvPatchScalarField.C
convectiveHeatFlux/convectiveHeatFluxFvPatchFields.C
fourier/fourierFvPatchScalarField.C
fourthOrderTimeScheme/fourthOrderDdt.C

LIB = $(FOAM_USER_LIBBIN)/libmyLib
```

Make/options

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = \
    -lfiniteVolume
```

ГРАНИЧНОЕ УСЛОВИЕ ПЕРВОГО РОДА

Данное ГУ варьирует значение по гармоническому закону с заданной длиной ряда,
амплитудами частотой и фазами.

Граничное условие реализовано для скалярных полей.

$$\psi = \sum_{k=1}^{N} A_k \cos\left(2\pi k \nu t + \theta_i\right)$$

Реализация ГУ содержит два файла — заголовочный (описание класса) и исходный код (определение методов класса)

ОПИСАНИЕ КЛАССА fourierFvPatchScalarField

```
class fourierFvPatchScalarField
:
    public fixedValueFvPatchScalarField
{
    // Private data

        //- expansion length
        label N_;

        //- amplitudes
        List<scalar> A_;

        //- phase shifts
        List<scalar> theta_;

        //- frequency
        scalar freq_;
```

```
public:

    //- Runtime type information
    TypeName("fourier");
```

ИНИЦИАЛИЗАЦИЯ ГРАНИЧНОГО УСЛОВИЯ

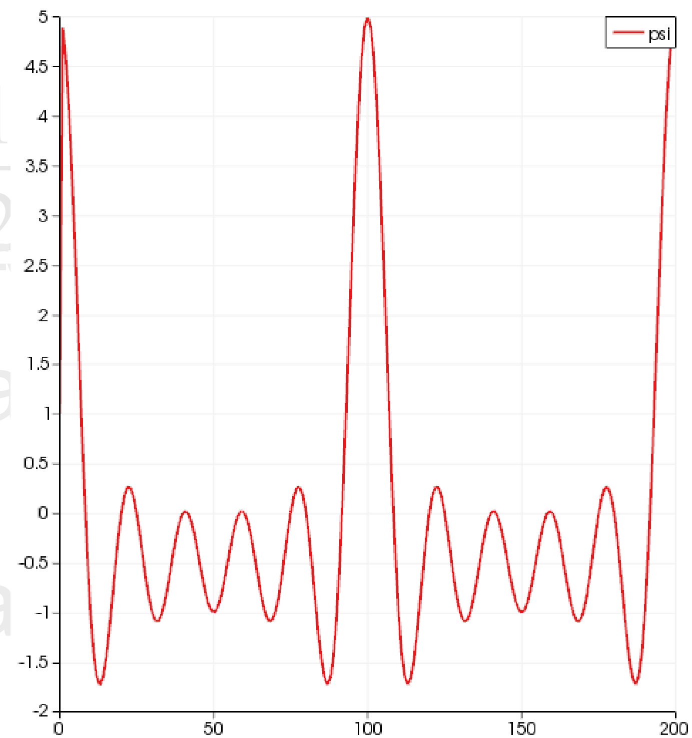
```
Foam::fourierFvPatchScalarField::fourierFvPatchScalarField
(
    const fvPatch& p, const DimensionedField<scalar, volMesh>& iF,
    const dictionary& dict
)
:
    fixedValueFvPatchScalarField(p, iF),
    N_(dict.lookupOrDefault<label>("N", 1)),
    A_(dict.lookupOrDefault<List<scalar> >("A", List<scalar>(1,1.0))),
    theta_(dict.lookupOrDefault<List<scalar> >("theta", List<scalar>(1,0.0))),
    freq_(dict.lookupOrDefault<scalar>("freq",1.0 ))
{
    if (dict.found("value"))
    {
        fvPatchField<scalar>::operator=
        (
            scalarField("value", dict, p.size())
        );
    }
    else
    {
        fvPatchField<scalar>::operator=(0.0);
    }
}
```

ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ПОЛЯ НА ГРАНИЦЕ

```
void Foam::fourierFvPatchScalarField::updateCoeffs()
{
    if (updated())
    {
        return;
    }

    scalar val = 0.0;
    scalar t = this->patch().boundaryMesh().mesh().time().value();
    forAll (A_, k)
    {
        val += A_[k] * cos (2*M_PI*scalar(k+1)*freq_*t + theta_[k]);
    }

    operator == (val);

    fixedValueFvPatchScalarField::updateCoeffs();
}
```

ПРОВЕРКА

- Создадим новый решатель с подключенной библиотекой myLib (см. ниже)
- И зададим 5 гармоник с частотой 2Гц и амплитудой 1



ГРАНИЧНОЕ УСЛОВИЕ ВТОРОГО РОДА

Данное ГУ вычисляет значение градиента по заданному потоку и коэффициенту диффузии.

Граничное условие реализовано для скалярных полей.

$$\frac{\partial \psi}{\partial \boldsymbol{n}} = -\frac{\boldsymbol{q}}{\lambda}$$

Реализация ГУ содержит два файла — заголовочный (описание класса) и исходный код (определение методов класса)

ОПИСАНИЕ КЛАССА conductiveHeatFluxFvPatchScalarField

```
class conductiveHeatFluxFvPatchScalarField
:
    public fixedGradientFvPatchScalarField
{
// Private data

    //- Heat flux [W]
    scalarField q_;

    //- wall heat exchange coefficient
    scalar lambdaWall_;

public:

    //- Runtime type information
    TypeName("conductiveHeatFlux");
```

ИНИЦИАЛИЗАЦИЯ ГРАНИЧНОГО УСЛОВИЯ

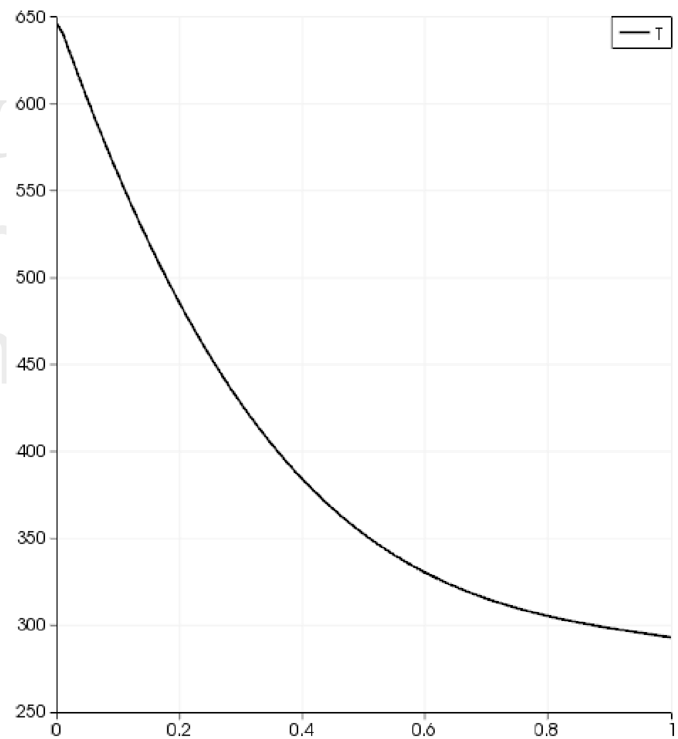
```
conductiveHeatFluxFvPatchScalarField::conductiveHeatFluxFvPatchScalarFiel
d
(const fvPatch& p, const DimensionedField<scalar, volMesh>& iF,
    const dictionary& dict)
    : fixedGradientFvPatchScalarField(p, iF), q_("q", dict, p.size()),
    lambdaWall_(dict.lookupOrDefault<scalar>("lambdaWall", 1e-6))
{
    if (dict.found("gradient")){
        gradient() = Field<scalar> ("gradient", dict, p.size());
    }
    else{
        gradient() = 0;
    }

    if (dict.found("value")){
        fvPatchField<scalar>::operator= (Field<scalar>("value", dict,
p.size()));
    }
    else{
        fvPatchField<scalar>::operator=(patchInternalField());
    }
}
```

ВЫЧИСЛЕНИЕ ГРАДИЕНТА ПОЛЯ НА ГРАНИЦЕ

```
void conductiveHeatFluxFvPatchScalarField::updateCoeffs()
{
    if (updated())
    {
        return;
    }

    gradient() = q_/ lambdaWall_;

    fixedGradientFvPatchScalarField::updateCoeffs();
}
```

ПРОВЕРКА

- Возьмем решатель parabFoam, подключим к нему библиотеку и проверим задачу распространения тепла при заданном потоке с одной стороны и фиксированной температуре с другой



ГРАНИЧНОЕ УСЛОВИЕ ТРЕТЬЕГО РОДА

$$\lambda \frac{\partial T}{\partial \boldsymbol{n}} + \alpha\left(T - T_{ref}\right) = 0$$

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

Данный тип ГУ основывает на смешанном ГУ, в котором используется заданное значение градиента и фиксированное значение.

f — грань, греческая гамма — доля вклада фиксированного значения / градиента в ГУ.

$$T_f = V_f \gamma + \left(1 - \gamma\right)\left(T_P + G_f / \Delta\right)$$

$$\frac{\lambda}{\alpha} \frac{T_f - T_P}{|x_f - x_P|} + T_f - T_{ref} = 0$$

$$T_f = \left(1 + \frac{\lambda}{\alpha}\frac{1}{\Delta}\right)^{-1}\left(T_{ref} + \frac{\lambda}{\alpha}\frac{1}{\Delta} T_P\right)$$

ЧИСЛЕННАЯ ДИСКРЕТИЗАЦИЯ ГУ РОБИНА В OPENFOAM

$$T_f = \left(1 + C\right)^{-1}\left(T_{ref} + C\,T_P\right) \quad C = \frac{\lambda}{\alpha}\frac{1}{\Delta} \quad 1 - \frac{1}{1+C} = \frac{C}{1+C}$$

$$T_f = V_f\,\gamma + \left(1-\gamma\right)\left(T_P + G_f/\Delta\right)$$

```
template<class Type>
void Foam::convectiveHeatFluxFvPatchField<Type>::updateCoeffs()
{
    if (this->updated())
    {
        return;
    }
    scalarField C = lambda_ / alpha_ * this->patch().deltaCoeffs();
    this->valueFraction() = 1 + 1./C;
    this->refValue() = Type(pTraits<Type>::one)*Tref_;

    this->refGrad() = Type(pTraits<Type>::zero);
    mixedFvPatchField<Type>::updateCoeffs();
}
```

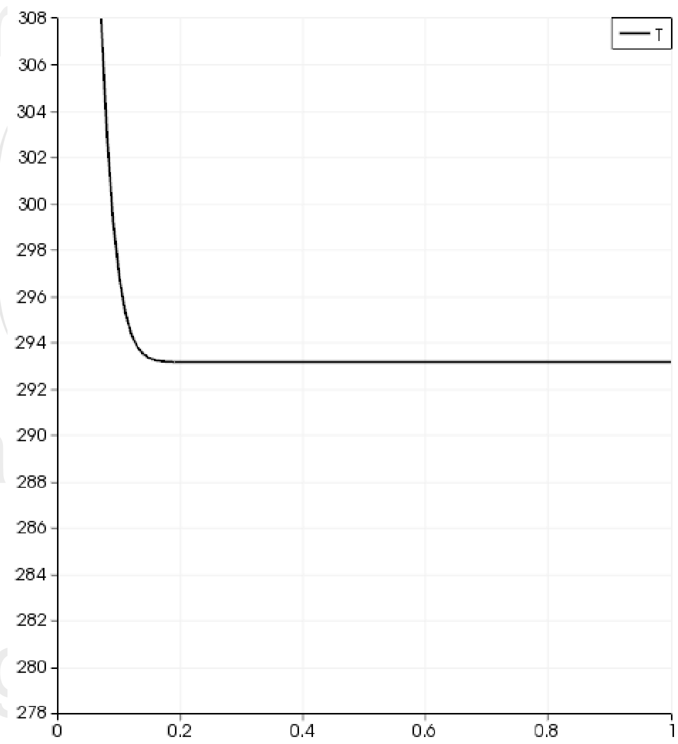
$$\gamma = 1 + C^{-1}$$

$$V_f = T_{ref}$$

$$G_f = 0$$

ПРОВЕРКА

- Как и в предыдущем примере, свяжем parabFoam с библиотекой и задачу распределения тепла в стержне, с одной стороны которого задано ГУ 3-го рода, с другой — 1-го

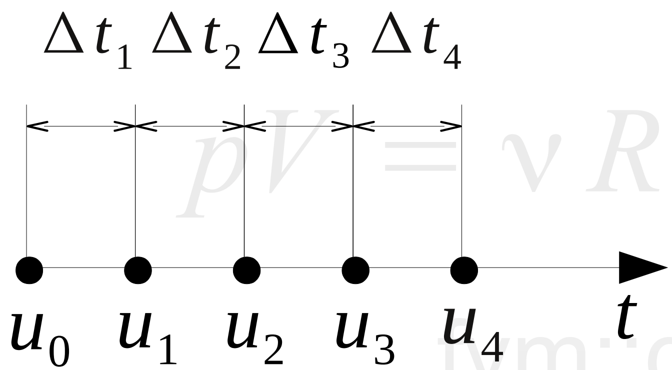


СВЯЗЫВАНИЕ БИБЛИОТЕКИ С РЕШАТЕЛЕМ

- Если необходимо вызывать функции классов непосредственно из решателя, то в options нужно добавить папки в которых выполняется поиск заголовочных файлов (раздел EXE_INC)
- В options в разделе EXE_LIBS необходимо указать расположение библиотеки и её имя
- Пример: myLibHyper1Foam

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude

EXE_LIBS = -lfiniteVolume \
        -L$(FOAM_USER_LIBBIN) -lmyLib
```

РАЗРАБОТКА НОВОЙ ЧИСЛЕННОЙ СХЕМЫ



$$\delta t_0 = \sum_{j=1}^{4} \Delta t_j \quad \delta t_1 = \sum_{j=2}^{4} \Delta t_j \quad \delta t_2 = \sum_{j=3}^{4} \Delta t_j$$

$$\delta t_3 = \sum_{j=3}^{4} \Delta t_j$$

$$u_0 \approx u_4 - \left(\frac{\partial u}{\partial t}\right)_{t_4} \delta t_0 + \frac{1}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_{t_4} (\delta t_0)^2 - \frac{1}{6}\left(\frac{\partial^3 u}{\partial t^3}\right)_{t_4} (\delta t_0)^3 + \frac{1}{24}\left(\frac{\partial^4 u}{\partial t^4}\right)_{t_4} (\delta t_0)^4$$

$$u_1 \approx u_4 - \left(\frac{\partial u}{\partial t}\right)_{t_4} \delta t_1 + \frac{1}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_{t_4} (\delta t_1)^2 - \frac{1}{6}\left(\frac{\partial^3 u}{\partial t^3}\right)_{t_4} (\delta t_1)^3 + \frac{1}{24}\left(\frac{\partial^4 u}{\partial t^4}\right)_{t_4} (\delta t_1)^4$$

$$u_2 \approx u_4 - \left(\frac{\partial u}{\partial t}\right)_{t_4} \delta t_2 + \frac{1}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_{t_4} (\delta t_2)^2 - \frac{1}{6}\left(\frac{\partial^3 u}{\partial t^3}\right)_{t_4} (\delta t_2)^3 + \frac{1}{24}\left(\frac{\partial^4 u}{\partial t^4}\right)_{t_4} (\delta t_2)^4$$

$$u_3 \approx u_4 - \left(\frac{\partial u}{\partial t}\right)_{t_4} \delta t_3 + \frac{1}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_{t_4} (\delta t_3)^2 - \frac{1}{6}\left(\frac{\partial^3 u}{\partial t^3}\right)_{t_4} (\delta t_3)^3 + \frac{1}{24}\left(\frac{\partial^4 u}{\partial t^4}\right)_{t_4} (\delta t_3)^4$$

РАЗРАБОТКА НОВОЙ ЧИСЛЕННОЙ СХЕМЫ (2)

Для данной системы уравнений требуется найти такие коэффициенты, которые бы позволяли найти аппроксимацию первой производной по времени с 4-м порядком точности (значения коэффициентов — на следующем слайде)

$$
\begin{pmatrix}
-\delta t_0 & \frac{1}{2}(\delta t_0)^2 & \frac{-1}{6}(\delta t_0)^3 & \frac{1}{24}(\delta t_0)^4 \\
-\delta t_1 & \frac{1}{2}(\delta t_1)^2 & \frac{-1}{6}(\delta t_1)^3 & \frac{1}{24}(\delta t_1)^4 \\
-\delta t_2 & \frac{1}{2}(\delta t_2)^2 & \frac{-1}{6}(\delta t_2)^3 & \frac{1}{24}(\delta t_2)^4 \\
-\delta t_3 & \frac{1}{2}(\delta t_3)^2 & \frac{-1}{6}(\delta t_3)^3 & \frac{1}{24}(\delta t_3)^4
\end{pmatrix}
\begin{pmatrix}
\left(\frac{\partial u}{\partial t}\right)_{t_4} \\
\left(\frac{\partial^2 u}{\partial t^2}\right)_{t_4} \\
\left(\frac{\partial^3 u}{\partial t^3}\right)_{t_4} \\
\left(\frac{\partial^4 u}{\partial t^4}\right)_{t_4}
\end{pmatrix}
=
\begin{pmatrix}
u_4 - u_0 \\
u_4 - u_1 \\
u_4 - u_2 \\
u_4 - u_3
\end{pmatrix}
$$

РАЗРАБОТКА НОВОЙ ЧИСЛЕННОЙ СХЕМЫ (3)

$$k_0=\frac{\delta t_1\delta t_2\delta t_3}{(-\delta t_3+\delta t_0)\delta t_0(-\delta t_1+\delta t_0)(-\delta t_2+\delta t_0)}$$

$$\left(\frac{\partial u}{\partial t}\right)_{t_4}=\sum_{j=0}^{k=3}k_j u_j-\sum_{j=0}^{k=3}k_j u_4$$

$$k_1=\frac{\delta t_0\delta t_2\delta t_3}{(\delta t_1-\delta t_3)(-\delta t_1+\delta t_0)\delta t_1(\delta t_1-\delta t_2)}$$

$$\frac{d}{dt}\int_V \rho V u\, dV\approx$$

$$k_2=\frac{\delta t_0\delta t_3\delta t_1}{(\delta t_2-\delta t_3)(\delta t_1\delta t_0-\delta t_2\delta t_0+(\delta t_2)^2-\delta t_2\delta t_1)\delta t_2}$$

$$\sum_{i=0}^{3}k_i\rho_i V_i u_i-\rho_4 V_4 u_4\sum_{i=0}^{3}k_i$$

$$k_3=\frac{\delta t_0\delta t_2\delta t_1}{(\delta t_0\delta t_2\delta t_1-\delta t_0\delta t_3\delta t_1+(\delta t_3)^2\delta t_0-\delta t_0\delta t_2\delta t_3+(\delta t_3)^2\delta t_1-\delta t_1\delta t_2\delta t_3-(\delta t_3)^3+(\delta t_3)^2\delta t_2)\delta t_3}$$

РАЗРАБОТКА НОВОЙ ЧИСЛЕННОЙ СХЕМЫ (4)

- Реализация схемы в OpenFOAM

```
const Foam::fvScalarMatrix& Foam::fourthOrderDdt::ddt()
{
    AdvanceInTime(); makeCoeffs();
    mtrx_ = tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            psi_,
            psi_.dimensions()*dimVol / dimTime
        )
    );
    forAll (mtrx_().diag(), i)
    {
        mtrx_().diag()[i] = - (k_[0] + k_[1] + k_[2] + k_[3]) * mesh_.V()[i];
        for (label j=0; j<=3; j++)
        {
            mtrx_().source()[i] -= k_[j]*oldV_[j][i]*oldPsi_[j][i];
        }
    }
    return mtrx_();
}
```

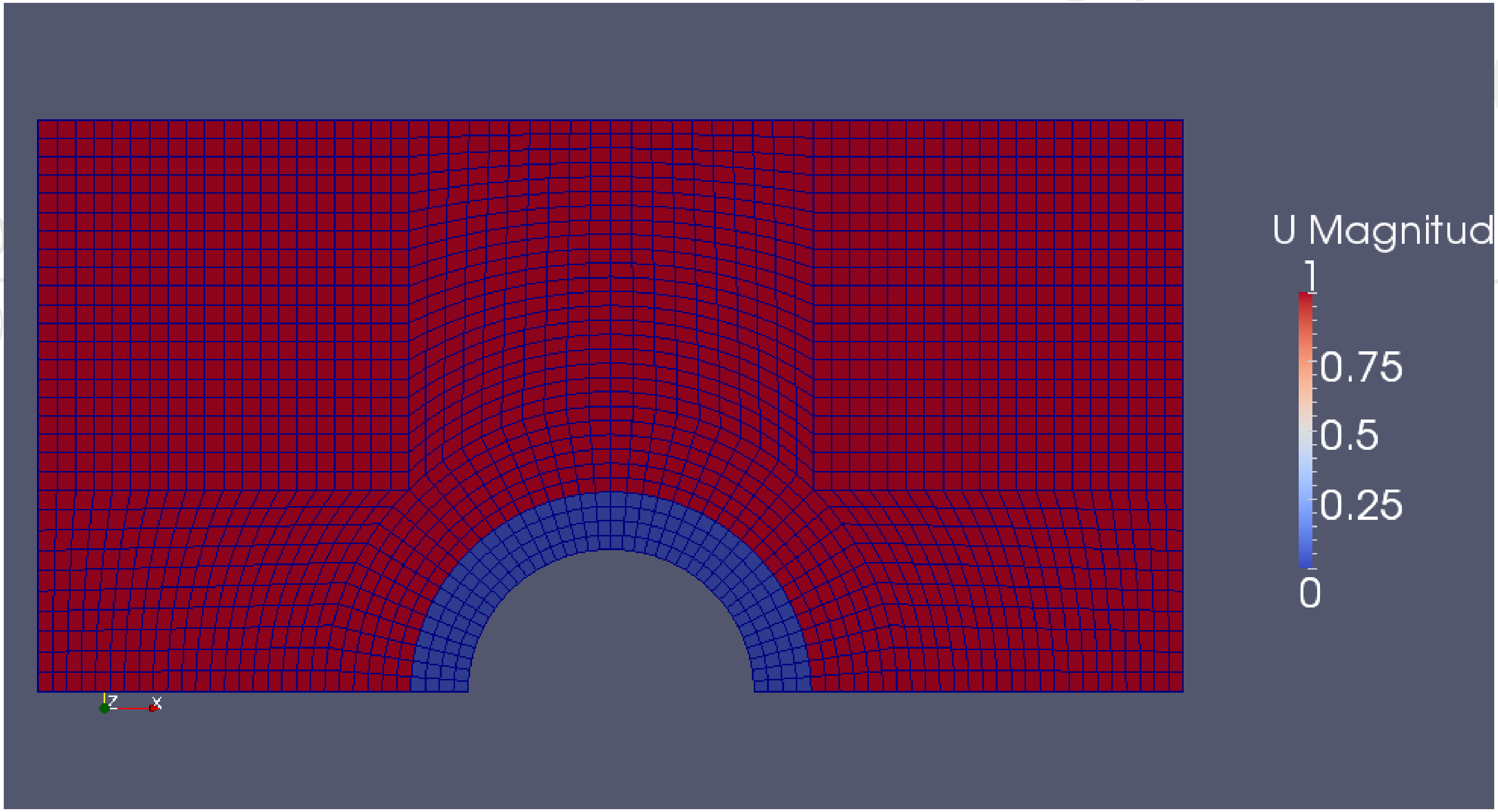
ПОДКЛЮЧЕНИ К РЕШАТЕЛЮ

- Воспользуемся решателем hyper1Foam
- Подключим библиотеку myLib
- Подключим её заголовочные файлы
- Создадим объект psiDdt класса fourthOrderDdt
- Изменим уравнение:

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

РАЗРАБОТКА УТИЛИТЫ ДЛЯ ПРЕПРОЦЕССИНГА

- Разработка утилиты, инициализирующее поле скорости в расчетной области так, что вдали от рассматриваемой границы вектор скорости равен одному значению, а вблизи нее - другому



ЗАДАНИЕ ЗНАЧЕНИЙ В ЦЕНТРАХ ОБЪЕМОВ

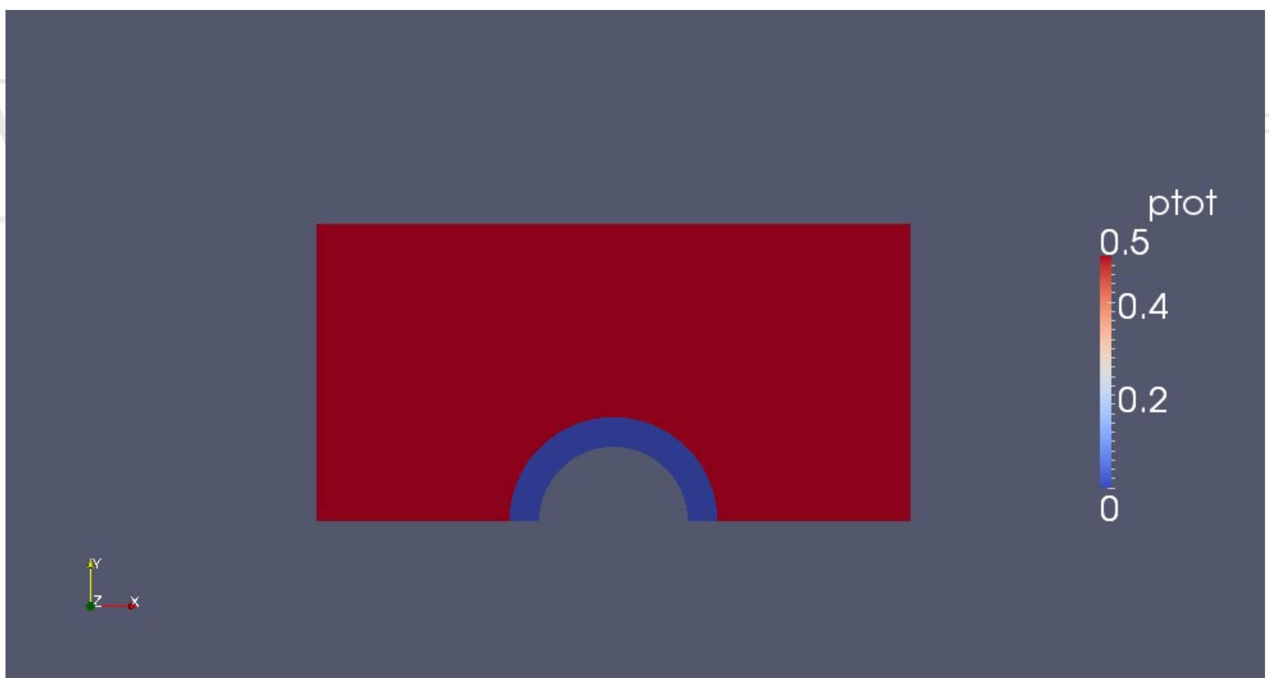
```
//set volume fields
forAll (mesh.C(), i)
{
    scalar dist = minPatchDist (mesh.C()[i], mesh, wallPatchId);
    if (dist > minDist)
    {
        U[i] = Ufar;
    }
    else
    {
        U[i] = Unear;
    }
}
```

ОПРЕДЕЛЕНИЕ МИНИМАЛЬНОГО РАССТОЯНИЯ

```
scalar minPatchDist (vector C, const fvMesh& mesh, label patchId)
{
    vectorField Cf = mesh.Cf().boundaryField()[patchId];
    scalar dist = 1 / VSMALL;
    forAll (Cf, i)
    {
        scalar cDist = mag(C - Cf[i]);
        if (cDist < dist)
        {
            dist = cDist;
        }
    }

    return dist;
}
```

РАЗРАБОТКА УТИЛИТЫ ДЛЯ ПОСТПРОЦЕССИНГА

- Требуется в каждой точке расчетной области (центрах контрольных объемов) вычислить полное давление как сумму избыточного и половины квадрата динамического



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)