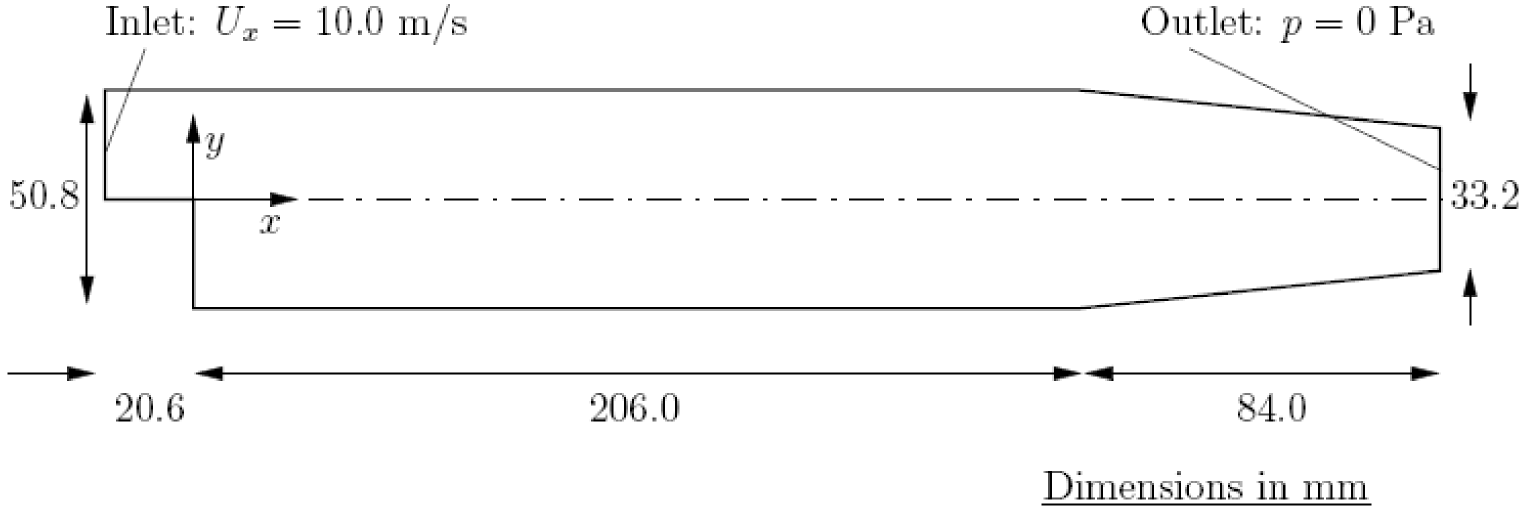
Rearward-Facing Step Flow

Sergei Strijhak

ISPRAS, Moscow

27.06.2016

Flow simulation in channel (pitzDaily case)
DOMAIN SIMULATION AND INITIAL DATA



U = 10 m/s on inlet, other patches - wall
URANS + K-e, K-omega SST turbulence model
LES (Large Eddy Simulation) + K one eddy equation model

Solvers: simpleFoam , pisoFoam

Using of Wall Functions

Goal: a) steady solution solver (…/tut/incompressible/simpleFoam)
b) unsteady solution solver (…/tut/incompressible/pisoFoam)

Experiment. Schlieren photograph. Re=22000

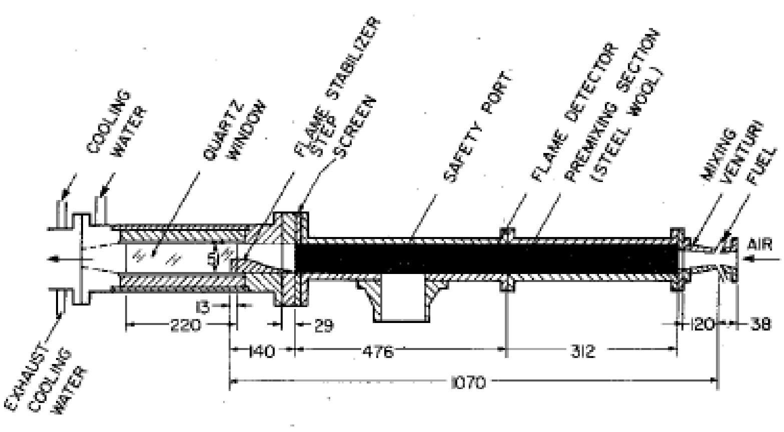


Fig. 1 Cross section of the two-dimensional combustor (all dimensions in mm).

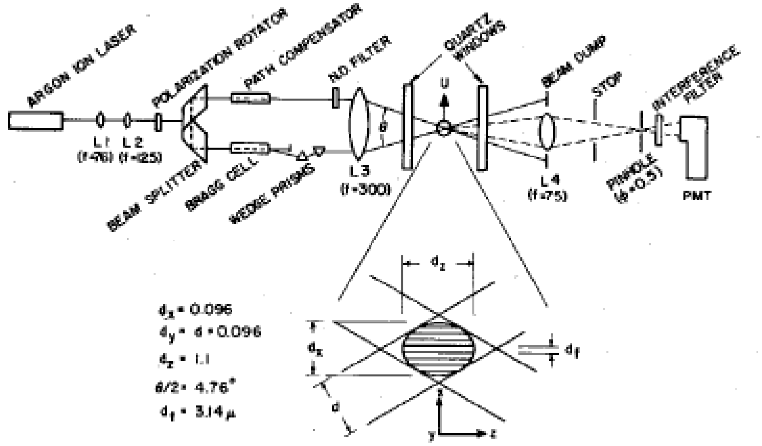


Fig. 2 Two-dimensional combustor LDV optics (all dimensions in mm).

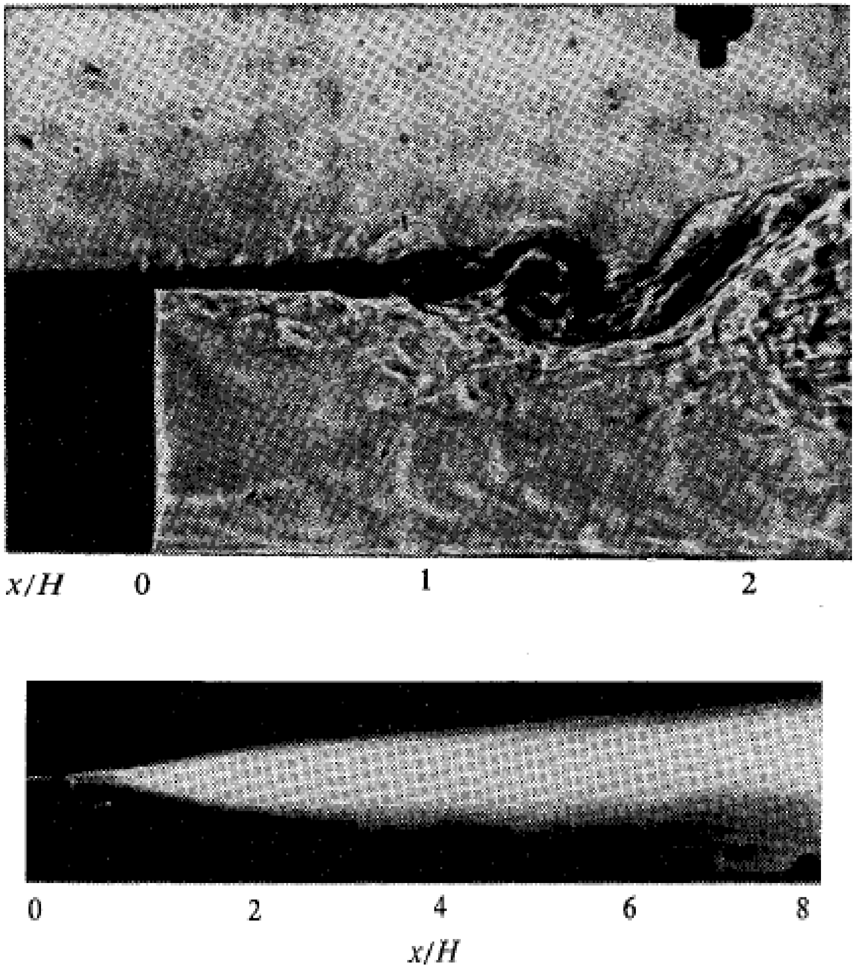
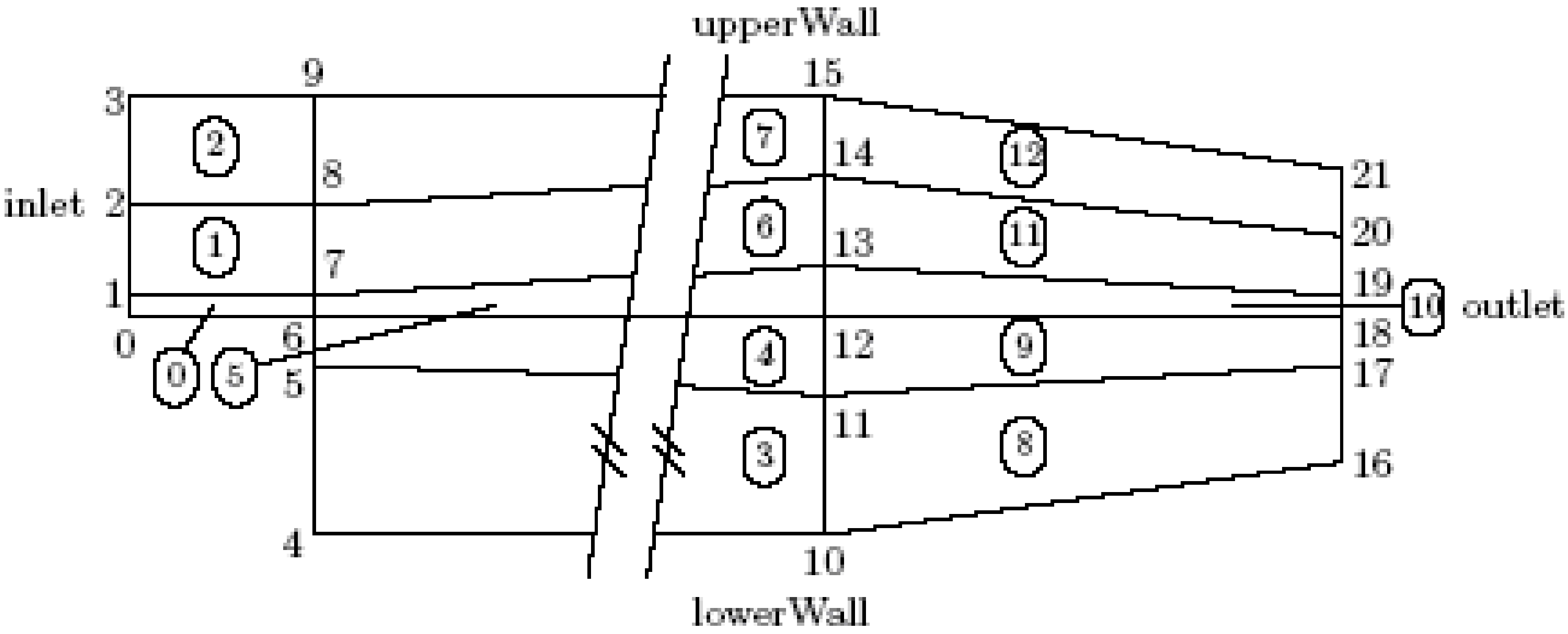


Fig. 4 Long exposure schlieren photograph of the reacting flow ($Re_H = 22{,}000$, $\phi = 0.57$, $\tau_{exp} = 33$ ms).

R.W. Pitz, J.W. Daily. AIAA Journal, 1983. vol.21. No. 11.

Block mesh

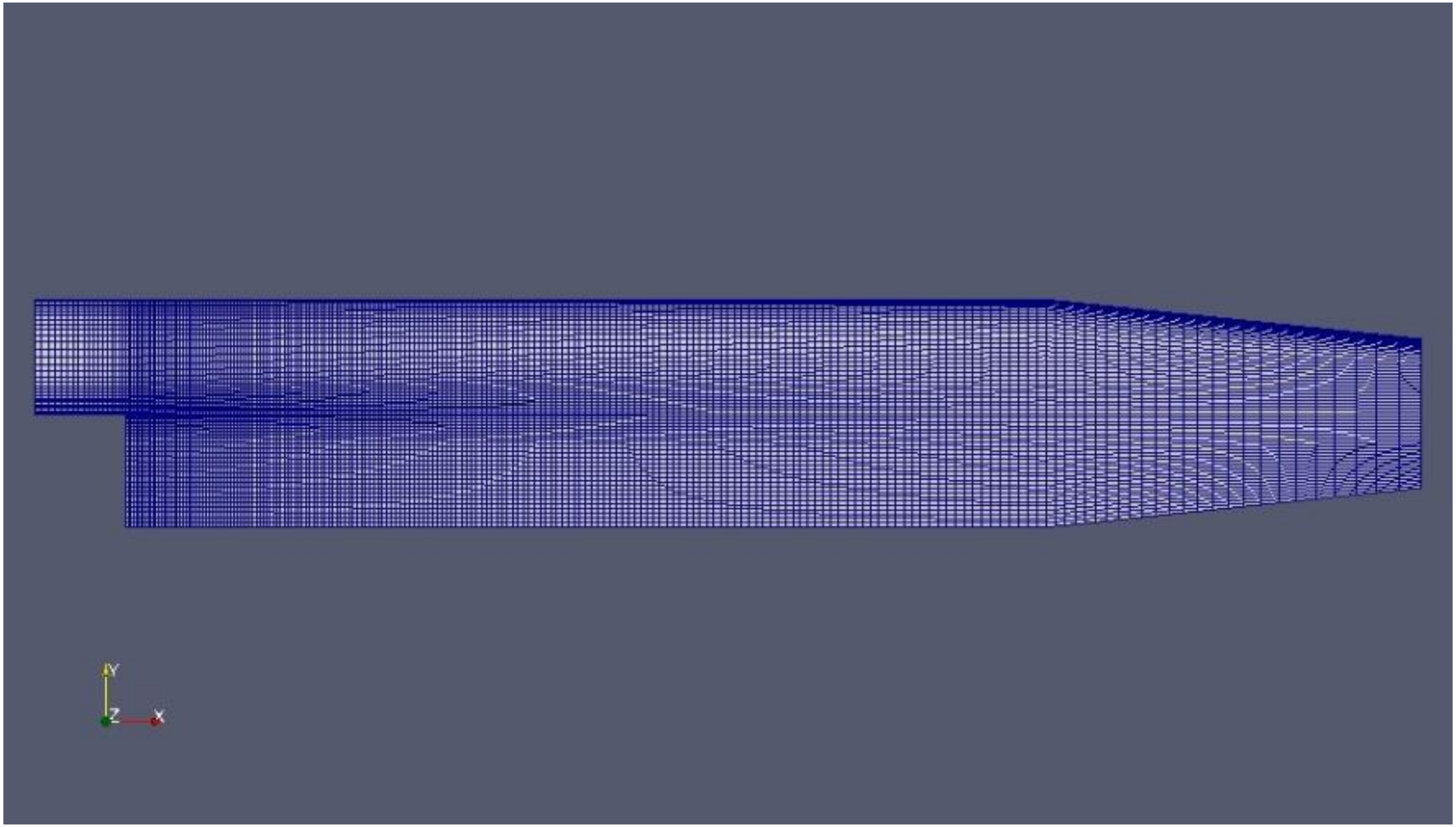


Block Mesh— file /constant/blockMeshDict
with 13 blocks
pitzDaily case

Fragments of BlockMeshDict file for PitzDaily case
13 different blocks

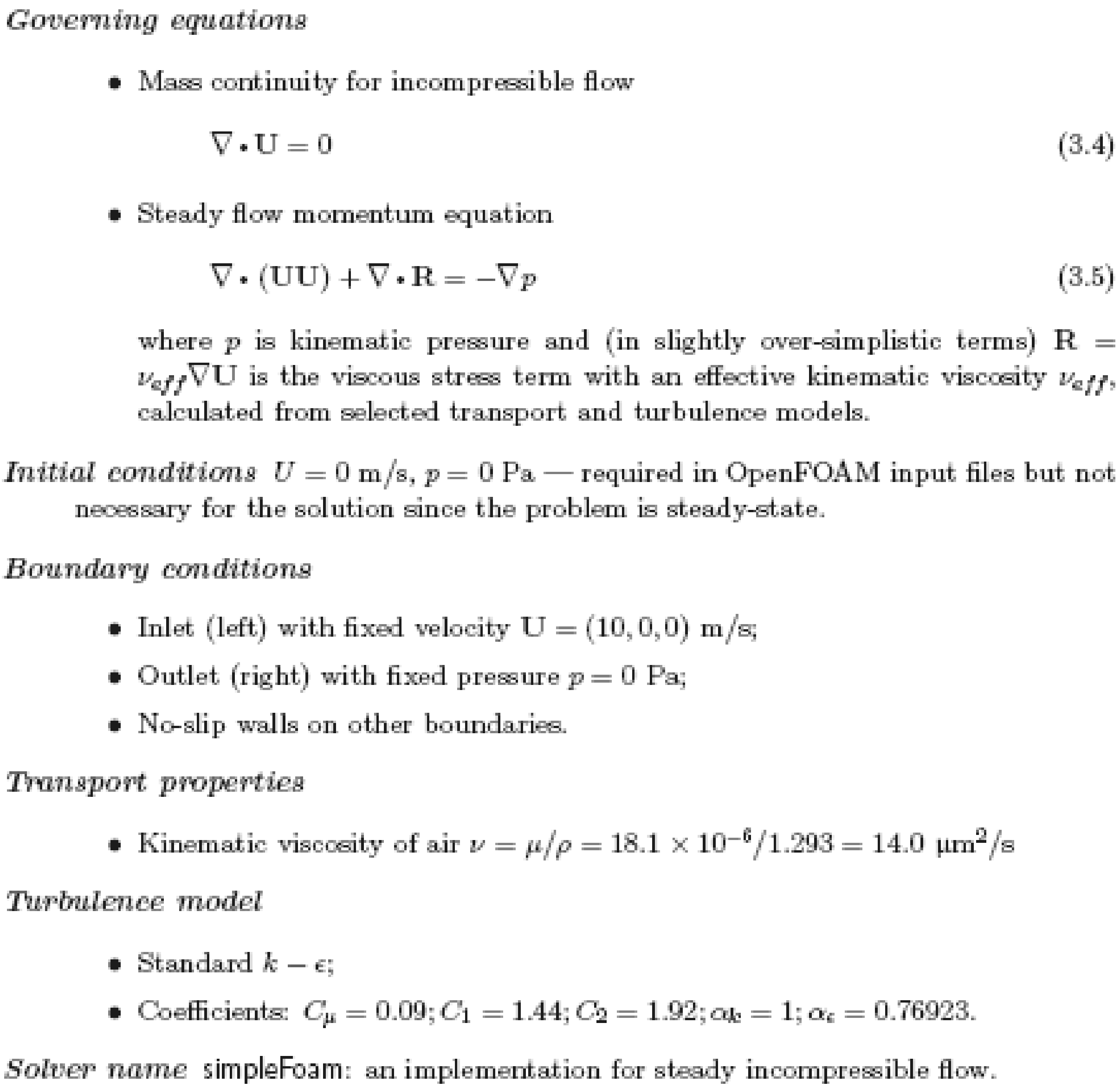
```
blocks
(
    hex (0 6 7 1 22 28 29 23) (18 7 1) simpleGrading (0.5 1.8 1)
    hex (1 7 8 2 23 29 30 24) (18 10 1) simpleGrading (0.5 4 1)
    hex (2 8 9 3 24 30 31 25) (18 13 1) simpleGrading (0.5 0.25 1)
    hex (4 10 11 5 26 32 33 27) (180 18 1) simpleGrading (4 1 1)
    hex (5 11 12 6 27 33 34 28) (180 9 1) edgeGrading (4 4 4 4 0.5 1 1 0.5 1 1 1 1)
    hex (6 12 13 7 28 34 35 29) (180 7 1) edgeGrading (4 4 4 4 1.8 1 1 1.8 1 1 1 1)
    hex (7 13 14 8 29 35 36 30) (180 10 1) edgeGrading (4 4 4 4 4 1 1 4 1 1 1 1)
    hex (8 14 15 9 30 36 37 31) (180 13 1) simpleGrading (4 0.25 1)
    hex (10 16 17 11 32 38 39 33) (25 18 1) simpleGrading (2.5 1 1)
    hex (11 17 18 12 33 39 40 34) (25 9 1) simpleGrading (2.5 1 1)
    hex (12 18 19 13 34 40 41 35) (25 7 1) simpleGrading (2.5 1 1)
    hex (13 19 20 14 35 41 42 36) (25 10 1) simpleGrading (2.5 1 1)
    hex (14 20 21 15 36 42 43 37) (25 13 1) simpleGrading (2.5 0.25 1)
);
```

Grid with refinement



2D case - 12225 cells,
3D case - 244500 cell (combustion/XiFoam/pitzDaily3D)

Mathematical model



Governing equations

- Mass continuity for incompressible flow

$$\nabla \bullet \mathrm{U} = 0 \tag{3.4}$$

- Steady flow momentum equation

$$\nabla \bullet (\mathrm{UU}) + \nabla \bullet \mathrm{R} = -\nabla p \tag{3.5}$$

where p is kinematic pressure and (in slightly over-simplistic terms) $\mathrm{R} = \nu_{eff} \nabla \mathrm{U}$ is the viscous stress term with an effective kinematic viscosity ν_{eff}, calculated from selected transport and turbulence models.

Initial conditions $U = 0$ m/s, $p = 0$ Pa — required in OpenFOAM input files but not necessary for the solution since the problem is steady-state.

Boundary conditions

- Inlet (left) with fixed velocity $\mathrm{U} = (10, 0, 0)$ m/s;
- Outlet (right) with fixed pressure $p = 0$ Pa;
- No-slip walls on other boundaries.

Transport properties

- Kinematic viscosity of air $\nu = \mu/\rho = 18.1 \times 10^{-6}/1.293 = 14.0\ \mu\mathrm{m}^2/\mathrm{s}$

Turbulence model

- Standard $k - \epsilon$;
- Coefficients: $C_\mu = 0.09; C_1 = 1.44; C_2 = 1.92; \alpha_k = 1; \alpha_\epsilon = 0.76923$.

Solver name `simpleFoam`: an implementation for steady incompressible flow.

Mathematical model
(URANS – Unsteady Reynolds Averaged Navier-Stokes)

$$\int_{V_P} \underbrace{\frac{\partial \rho\phi}{\partial t} dV}_{\text{temporal derivative}} + \int_{V_P} \underbrace{\nabla\cdot(\rho \mathbf{u}\phi)\,dV}_{\text{convective term}} - \int_{V_P}\underbrace{\nabla\cdot(\rho\Gamma_\phi\nabla\phi)\,dV}_{\text{diffusion term}} = \int_{V_P}\underbrace{S_\phi(\phi)\,dV}_{\text{source term}} \qquad (1)$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho u_i k)}{\partial x_i} = \tilde{P}_k - \beta^* \rho k \omega + \frac{\partial}{\partial x_i}\left[(\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_i}\right],$$

k-omega SST model of turbulence

$$\frac{\partial(\rho\omega)}{\partial t} + \frac{\partial(\rho u_i \omega)}{\partial x_i} = \alpha\rho S^2 - \beta\rho\omega^2 + \frac{\partial}{\partial x_i}\left[(\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_i}\right] + 2(1 - F_1)\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \qquad (2)$$

$$\frac{\partial(\rho\tilde{v})}{\partial t} + \frac{\partial}{\partial x_j}(\rho\tilde{v}u_j) = \frac{1}{\sigma_{\tilde{v}}}\left\{\frac{\partial}{\partial x_j}\left[(\mu + \rho\tilde{v})\frac{\partial \tilde{v}}{\partial x_j}\right] + C_{b2}\rho\left(\frac{\partial \tilde{v}}{\partial x_j}\right)^2\right\} + G_v - Y_v; \qquad (3)$$

Spalart-Allmars model of turbulence

Definition of problem: Initial and BC conditions, numerical schemes and linear solvers

Математическая модель расчета параметров течения
(LES – Large Eddy Simulation)

$$u = \bar{u} + \bar{u}' \qquad \bar{u} = \int_D G(\zeta, \Delta)\, u(\zeta, t)\, d^3\zeta$$

$$\Delta = V^{1/3} = (\Delta x \Delta y \Delta z)^{1/3}$$

$$\partial_t \bar{u} + \nabla \cdot (\bar{u} \otimes \bar{u}) = \nabla \cdot (\bar{S} - B) \qquad S = -pI + 2\nu D$$

$$D = 0.5(\nabla u + \nabla u^T) \qquad B = L + C + R$$

Дифференциальное уравнение для подсеточной кинетической энергии:

$$\frac{\partial K}{\partial t} + \nabla \cdot (K\bar{U}) = \nabla\left[(\nu + \nu_{SGS}) \cdot \nabla K\right] - \varepsilon - \tau \cdot \bar{S}$$

Метод конечного объема

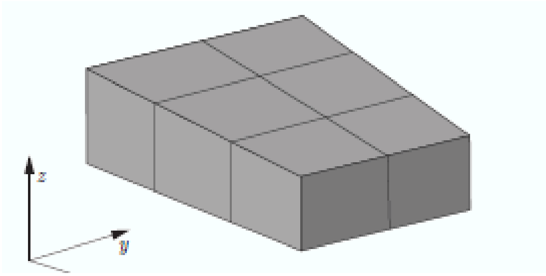


Рис. 31. Расчетная область

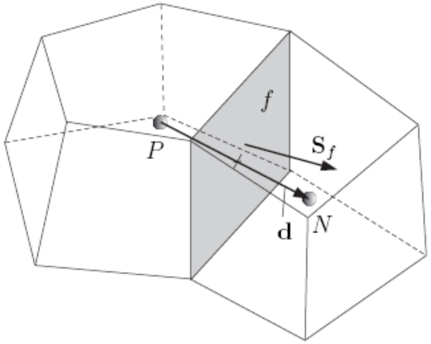


Рис. 32. Соседние расчетные ячейки

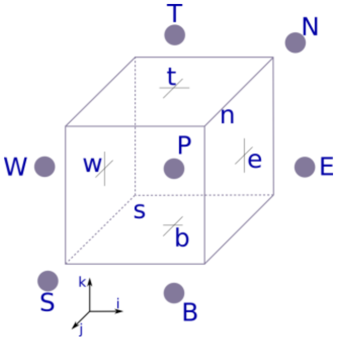


Рис. 33. Квадратная ячейка

- Декомпозиция расчетной области на ячейки
- Запись законов сохранения для каждой ячейки в интегральной форме
- Разложение функции в ряд Тейлора
- Аппроксимация интегралов по объему и поверхности
- Формула Остроградского-Гаусса и теорема о среднем
- Линейная интерполяция функции для нахождения значений в центре грани
- Итерационный метод решения СЛАУ (Метод бисопряженных градиентов PCG, PBiCG с предобуславливателем DILU)
- Задание невязок и проведение расчета

Вспомогательные вычисления

Постулирование линейного изменения переменной ϕ во времени.
Разложение функции в ряд Тейлора:

$$\phi\left(t+\Delta t\right)=\phi^{t}+\Delta t\left(\frac{\partial\phi}{\partial t}\right)^{t}+\mathrm{O}\left(\Delta t^{2}\right)$$

Пространственное изменение переменной. Возможен второй порядок точности по пространству и времени:

$$\phi\left(x\right)=\phi_{p}+\left(x-x_{p}\right)\cdot\left(\nabla\phi\right)_{P}+\mathrm{O}\left(\left\|\left(x-x_{p}\right)\right\|^{2}\right)$$

Аппроксимация объемного интеграла:

$$\int_{V}\phi dV=\int_{V}\left[\phi_{P}+\left(x-x_{P}\right)\cdot\left(\nabla\phi\right)_{P}\right]dV$$

$$=\phi_{P}\int_{V}dV+\left(\nabla\phi\right)_{P}\cdot\int_{V}\left(x-x_{P}\right)dV=\phi_{P}V_{P}$$

Аппроксимация поверхностного интеграла:

$$\oint_{S}n\,\phi\,dS=\sum_{f}\int_{S_f}n\,\phi_{f}\,dS_{f}=\sum_{f}\int_{S_f}n\left[\phi_{f}+\left(x-x_{f}\right)\cdot\left(\nabla\phi\right)_{f}\right]dS_{f}=\sum_{f}s_{f}\phi_{f}$$

Алгебраический аналог уравнений

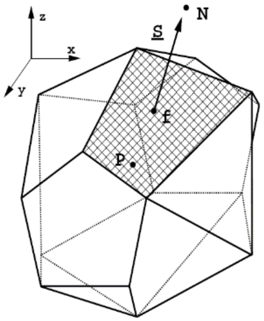


Рис. 34. Произвольная ячейка

Рис. 35. 1D расчетная ячейка

$$a_P \phi_P - \sum_F a_F \phi_F = b_P$$

$$F = \{E, W\}$$

$$\frac{d}{dx}\phi = C_e \phi_e - C_w \phi_w \qquad C_e = (\rho u A)_e \qquad C_w = (\rho u A)_w$$

$$-\frac{d}{dx}\Gamma\frac{d}{dx}\phi = -\left[D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W)\right] \quad D_e = \left(\frac{\Gamma A}{\Delta x}\right)_e \quad D_w = \left(\frac{\Gamma A}{\Delta x}\right)_w$$

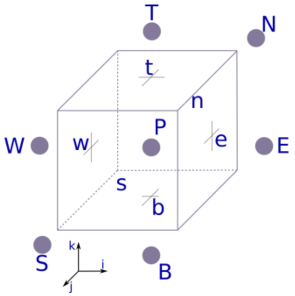


Рис. 36. Квадратная ячейка

$$a_P \phi_P - \sum_F a_F \phi_F = b_P$$

$$F = \{E, W, N, S, T, B\}$$

СЛАУ решается итерационными методами:

- Метод сопряженных градиентов
- Многосеточный метод
- Метод Гаусса-Зейделя

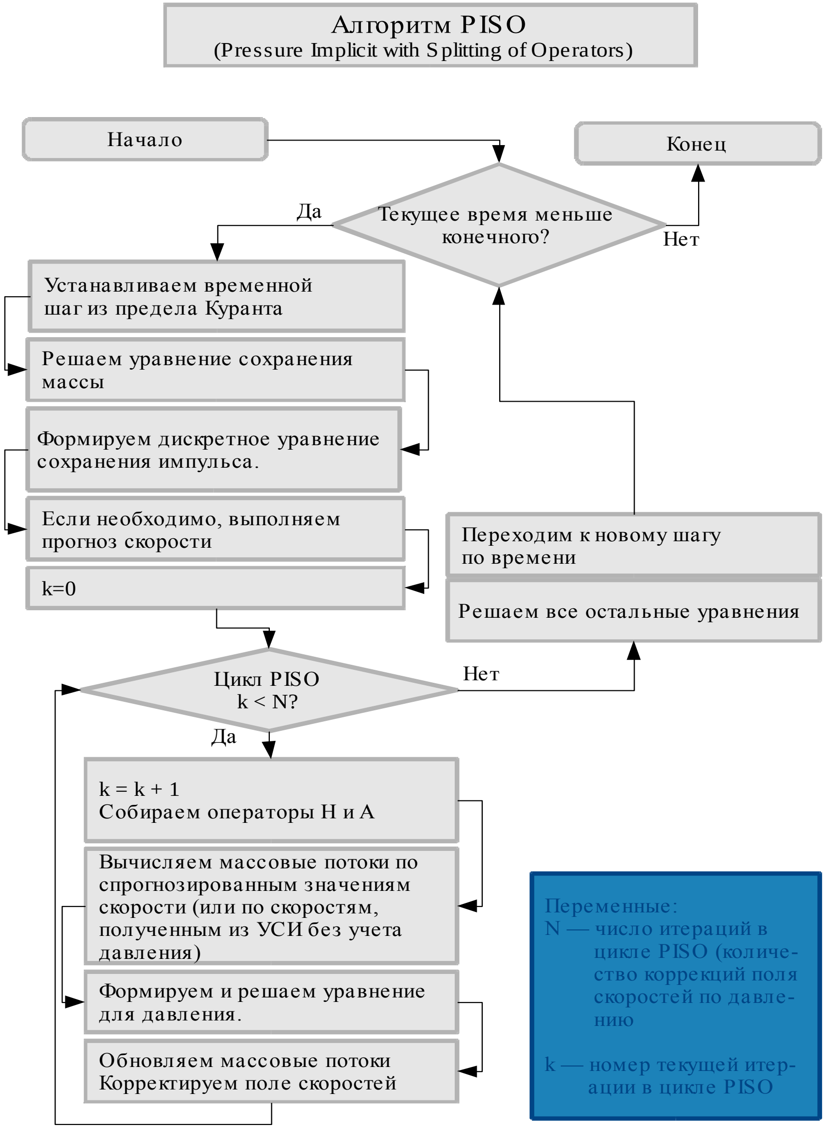
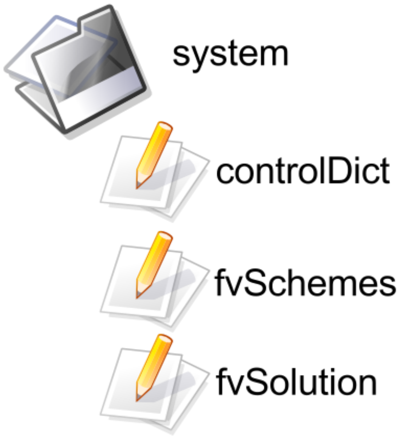


Рис. 37. Схема алгоритма PISO

Initial data for pitzDaily (tutorial case incompressible/simpleFoam). Folder of case:

```
[cfd1@master simpleFoam]$ cd pitzDailyParallel/
[cfd1@master pitzDailyParallel]$ ll
total 12
drwxr-xr-x 2 cfd1 sm3 4096 Dec 22 16:43 0
drwxr-xr-x 3 cfd1 sm3 4096 Dec 22 16:43 constant
drwxr-xr-x 2 cfd1 sm3 4096 Dec 22 16:48 system
[cfd1@master pitzDailyParallel]$
[cfd1@master pitzDailyParallel]$ cd system/
[cfd1@master system]$ ll
total 16
-rw-r----- 1 cfd1 sm3 1222 Dec 22 16:43 controlDict
-rw-r----- 1 cfd1 sm3 1206 Dec 22 16:48 decomposeParDict
-rw-r----- 1 cfd1 sm3 1877 Dec 22 16:43 fvSchemes
-rw-r----- 1 cfd1 sm3 1940 Dec 22 16:43 fvSolution
```



Data for velocity. File 0/U

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{    inlet           {       type            fixedValue;       value           uniform (10 0 0);    }

outlet           {       type            zeroGradient;    }

upperWall        {       type            fixedValue;       value           uniform (0 0 0);    }

lowerWall        {       type            fixedValue;       value           uniform (0 0 0);    }

frontAndBack     {       type            empty;    }}//
************************************************************************* //
```

Data for pressure. File: 0/P

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {  type            zeroGradient; }

    outlet
    { type            fixedValue;
      value           uniform 0; }

    upperWall
    { type            zeroGradient; }

    lowerWall
    { type            zeroGradient; }

    frontAndBack
    { type            empty;}
}
```

Data for turbulent kinetic energy. File: 0/k

Using of wall function: kqRWallFunction

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.375;

boundaryField
{
inlet    {       type            fixedValue;       value           uniform 0.375;    }
outlet    {       type            zeroGradient;    }
upperWall    {       type            kqRWallFunction;       value           uniform 0.375;    }
lowerWall    {       type            kqRWallFunction;       value           uniform 0.375;    }
frontAndBack    {       type            empty;
 }
}
// ************************************************************************* //
```

Data for energy dissipation. File: 0/epsilon

Using of wall function: epsilonWallFunction

```
dimensions      [0 2 -3 0 0 0 0];
internalField   uniform 14.855;
boundaryField
{
    inlet
    {
        type            fixedValue;
        value            uniform 14.855; }
    outlet
    {
        type            zeroGradient;}
    upperWall
    {
        type            epsilonWallFunction;
        value            uniform 14.855; }
    lowerWall
    {
        type            epsilonWallFunction;
        value            uniform 14.855; }
    frontAndBack
    {
        type            empty;}
}
```

Data for Reynolds stress tensor. File: 0/R

Using of wall function: kqRWallFunction

```
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform (0 0 0 0 0 0);
boundaryField
{
    inlet
    {
        type            fixedValue;
        value            uniform (0 0 0 0 0 0); }

    outlet
    {
        type            zeroGradient; }
    upperWall
    {
        type            kqRWallFunction;
        value            uniform ( 0 0 0 0 0 0 ); }
    lowerWall
    {
        type            kqRWallFunction;
        value            uniform ( 0 0 0 0 0 0 ); }
    frontAndBack
    {
        type            empty; }
}
```

Data for viscosity. File: 0/nut

Using of wall function: nutWallFunction

```
dimensions   [0 2 -1 0 0 0 0];
internalField  uniform 0;
boundaryField
{
    inlet
    {
        type      calculated;
        value      uniform 0;}
    outlet
    {
        type      calculated;
        value      uniform 0;}
    upperWall
    {
        type      nutWallFunction;
        value      uniform 0;}
    lowerWall
    {
        type      nutWallFunction;
        value      uniform 0;}
    frontAndBack
    {
        type      empty;}
}
```

File for data simulation /system/controlDict

```
application     simpleFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         10;
deltaT          1;
writeControl    timeStep;
writeInterval   1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
```

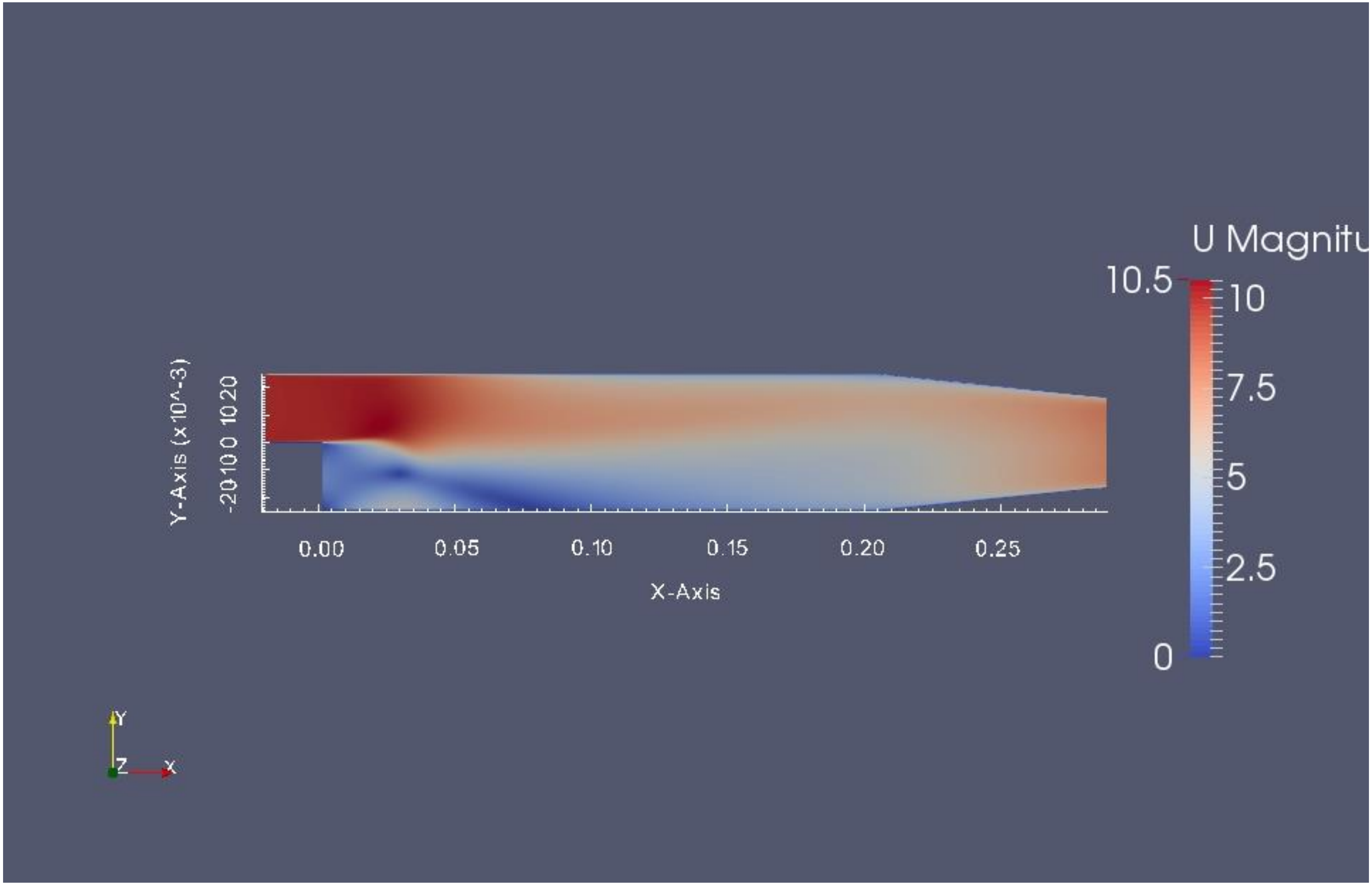
File for numerical schemes /system/fvSchemes

```
gradSchemes{    default         Gauss linear;
grad(p)         Gauss linear;
grad(U)         Gauss linear;}
divSchemes{    default         none;
div(phi,U)      Gauss GammaV 1.0;
div(phi,k)      Gauss Gamma 1.0;
div(phi,epsilon) Gauss Gamma 1.0;
div(phi,omega)  Gauss Gamma 1.0;
 div(phi,R)      Gauss Gamma 1.0;
div(R)          Gauss linear;
div(phi,nuTilda) Gauss upwind;
div((nuEff*dev(grad(U).T()))) Gauss linear;}
laplacianSchemes{    default         none;
laplacian(nuEff,U) Gauss linear corrected;
laplacian((1|A(U)),p) Gauss linear corrected;
laplacian(DkEff,k) Gauss linear corrected;
laplacian(DepsilonEff,epsilon) Gauss linear corrected;
 laplacian(DomegaEff,omega) Gauss linear corrected;
laplacian(DREff,R) Gauss linear corrected;
laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;}
interpolationSchemes{    default         linear;    interpolate(U)  linear;}
snGradSchemes{    default         corrected;}
fluxRequired{    default         no;    p;}
```

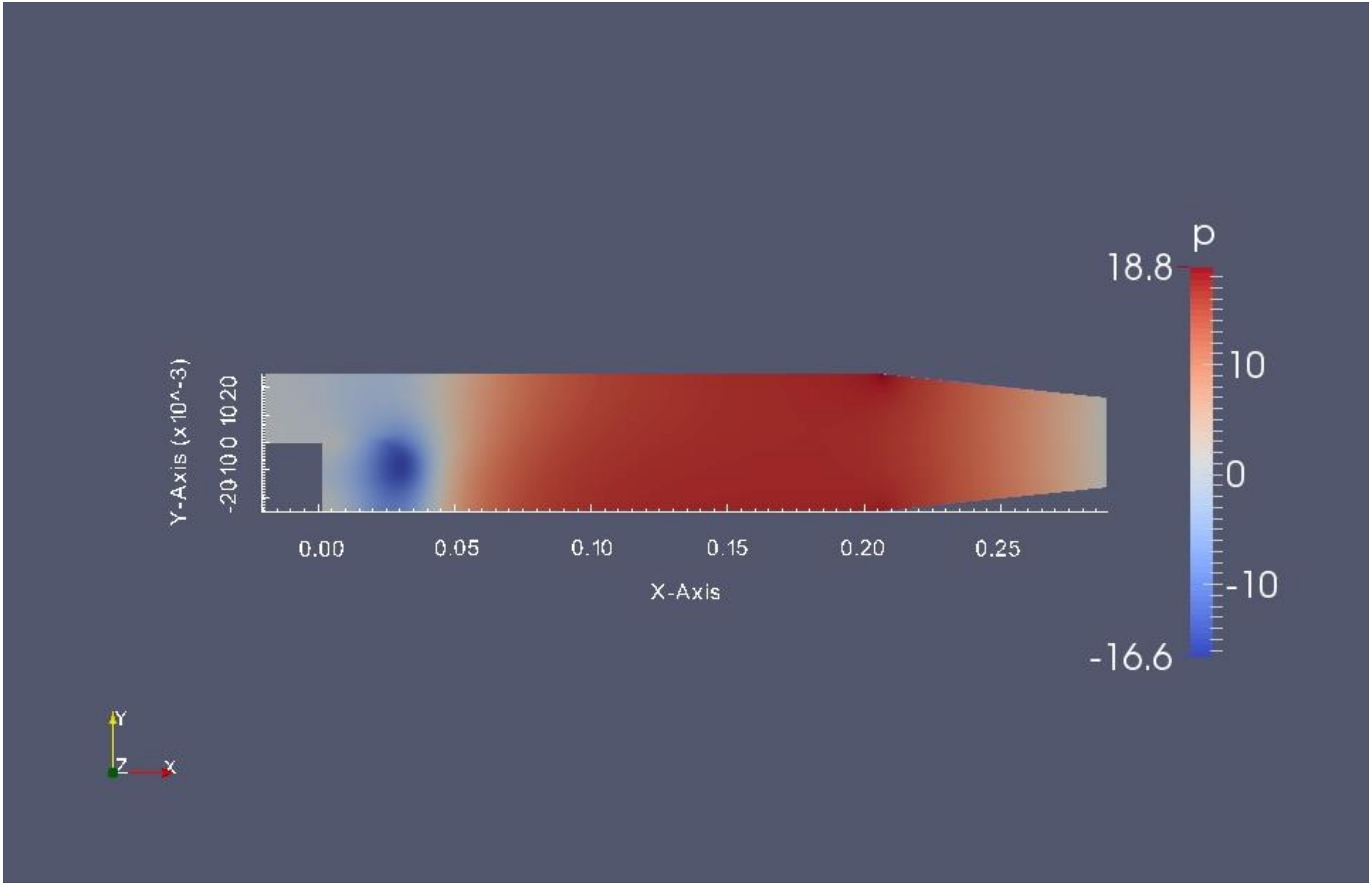
File for linear system solvers: /system/fvSolutions

```
FoamFile
{    version     2.0;    format      ascii;    class       dictionary;    object      fvSolution;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

Solvers {

p PCG    {       preconditioner   DIC;       tolerance       1e-06;       relTol          0.01;    };
U PBiCG    {       preconditioner   DILU;       tolerance       1e-05;       relTol          0.1;    };
k PBiCG    {       preconditioner   DILU;       tolerance       1e-05;       relTol          0.1;    };
epsilon PBiCG    {       preconditioner   DILU;       tolerance       1e-05;       relTol          0.1;    };
omega PBiCG    {       preconditioner   DILU;       tolerance       1e-05;       relTol          0.1;    };
R PBiCG    {       preconditioner   DILU;       tolerance       1e-05;       relTol          0.1;    };
nuTilda PBiCG    {       preconditioner   DILU;       tolerance       1e-05;       relTol          0.1;    };}

SIMPLE
{    nNonOrthogonalCorrectors 0;}
relaxationFactors
{    p              0.3;    U              0.7;    k              0.7;    epsilon        0.7;    omega          0.7;    R
0.7;    nuTilda        0.7;}
PISO
{    nCorrectors     4;    nNonOrthogonalCorrectors 0;    pRefCell        0;    pRefValue       0;}
// *************************************************************************** //
```

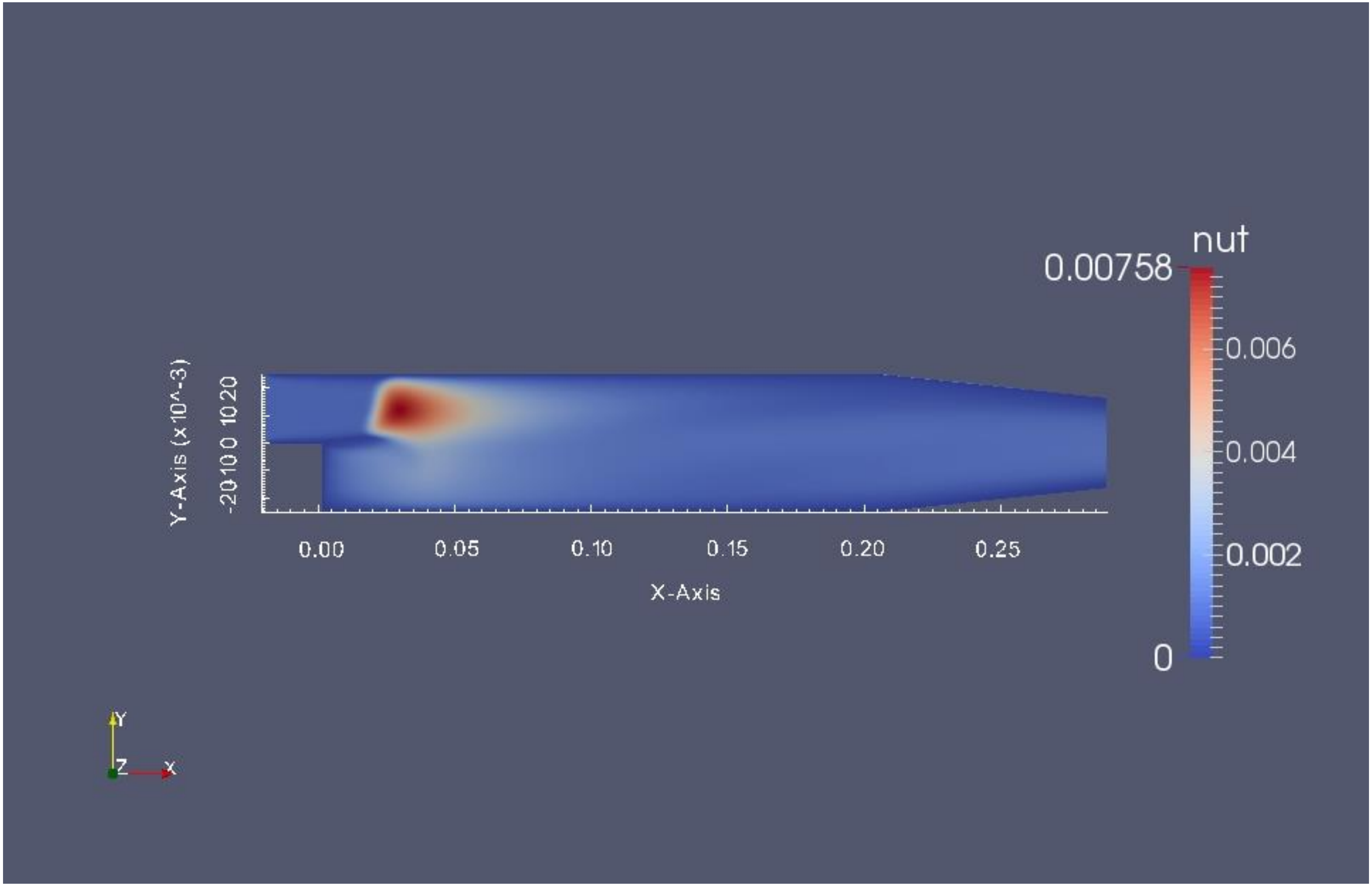
Results of UMagnitude with simpleFoam. T=100 s.



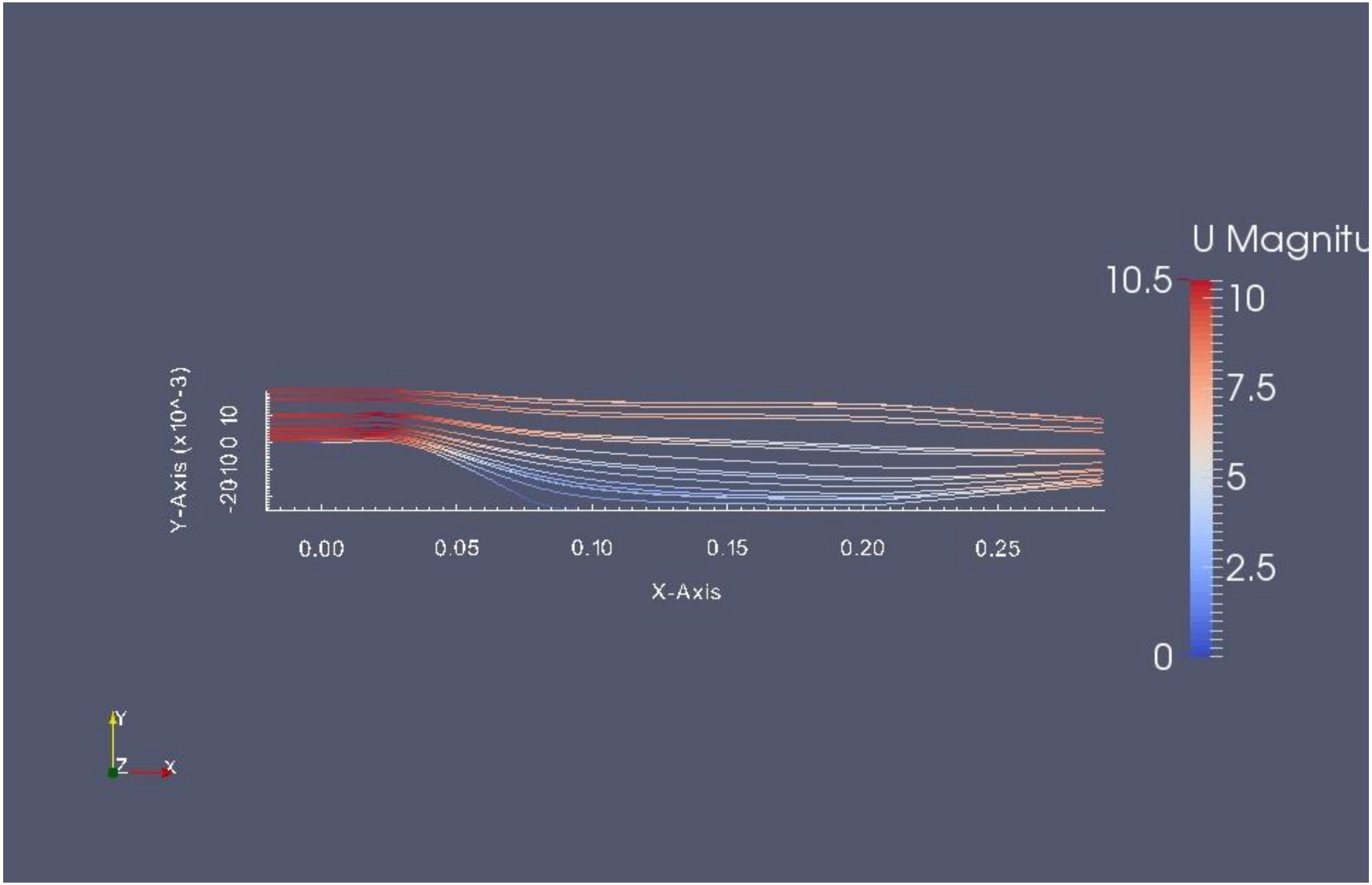
Results of pressure p in simpleFoam. T=100 s.



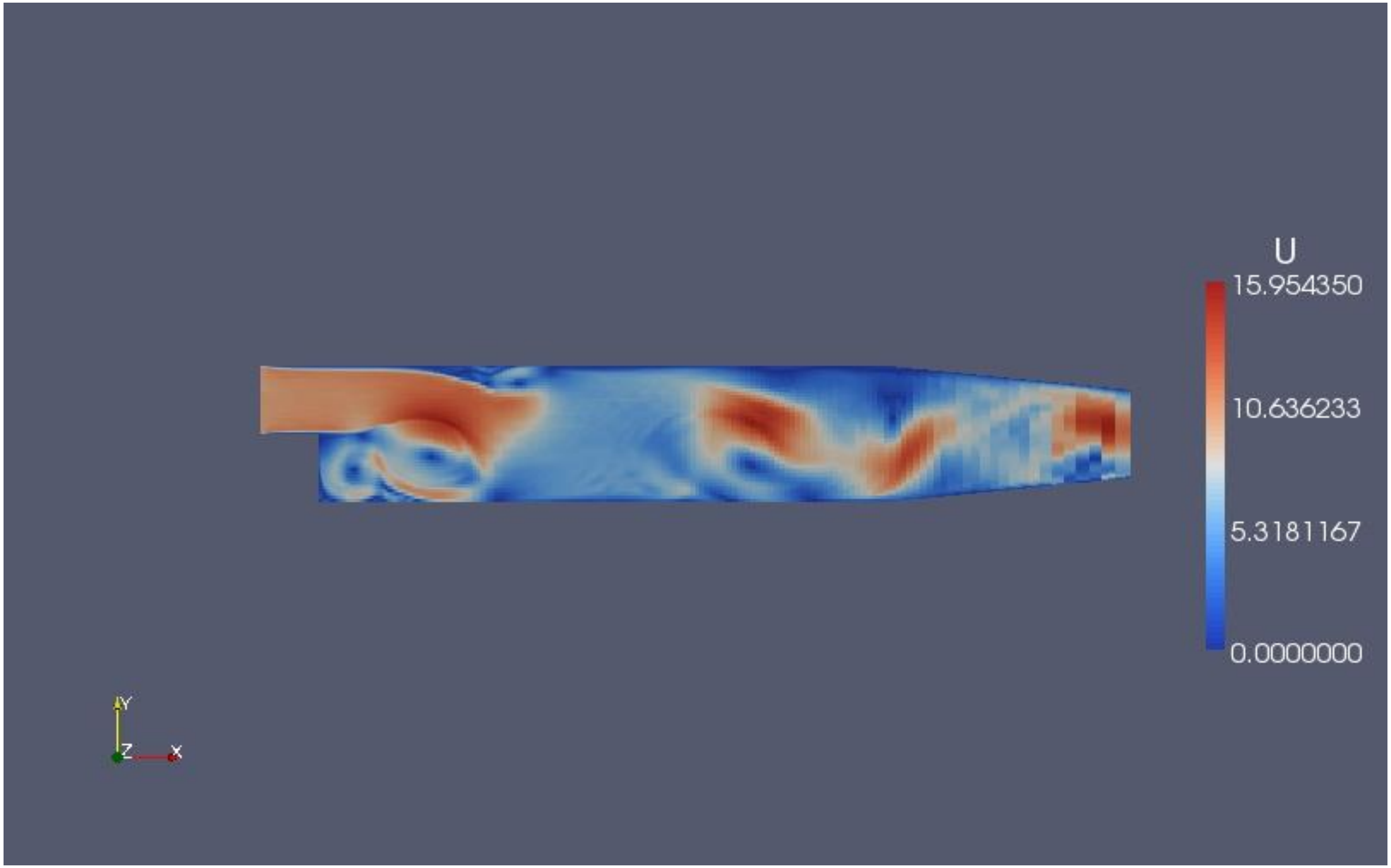
Results of nut with simpleFoam. T=100 s.



Results of stream functions with simpleFoam. T=100 s.



Results of UMagnitude with pisoFoam



LES model. One eddy equation.

Algorithm for each case in OpenFOAM

$ blockMesh – mesh preparation

$ checkMesh – check mesh

$ simpleFoam ($pisoFoam) – run of solver

$ yPlusRAS - definition of yPlus field for URANS plus turbulence model

$ foamToVTK - translation of results to VTK format

$ touch 1.foam – creation of file ‘foam’ format

$ simpleFoam > log & - creation of log file

$ foamLog log – script, using Linux commands grep, awk , sed, for data analysis.

$ gnuplot – for graphics creation

Start Paraview and load VTL files for data analysis.