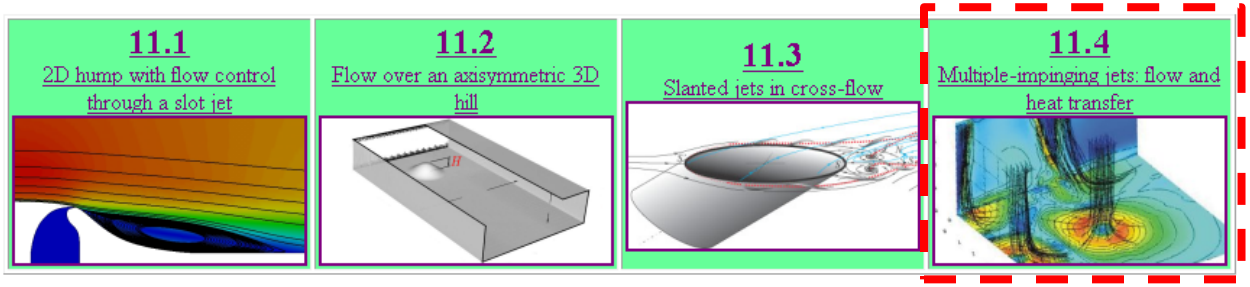
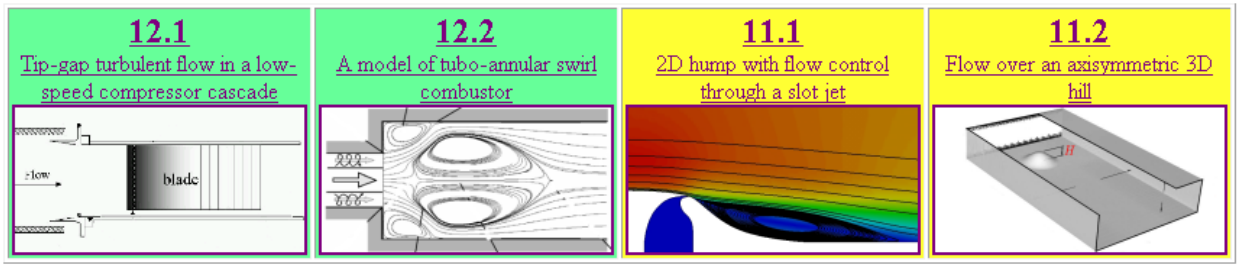
・Ercoftac SIG15 testcase ベンチマークの進捗

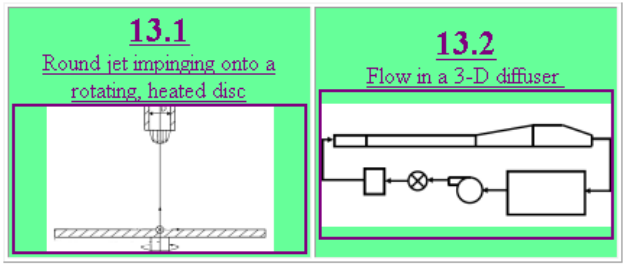
11th workshop at Chalmers University of Technology (7-8 April 2005)



12th workshop at Technical University of Berlin (12-13 October 2006)



13th workshop at Graz University of Technology (25-26 September 2008)



・Ercoftac SIG15 testcase　case11.4

Modelling and calculation of flow and
heat transfer in multiple impinging jets

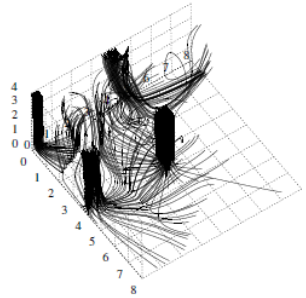
PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 9 december 2003 om 15.30 uur

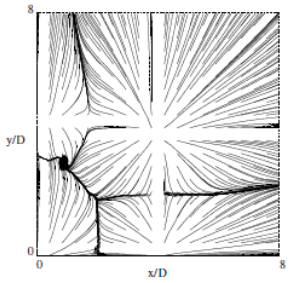
door Lukas THIELEN
ingenieur luchtvaart en ruimtevaart
geboren te Maasbree

デルフト工科大 Hanjalic研究室が
取り組んだ事例

加熱壁面への衝突流
吹き出し口の形状によっては
対称条件で解析しても、結果は
非対称なものになる

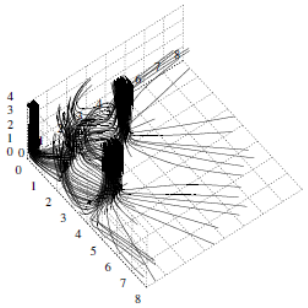


(a) the computational domain

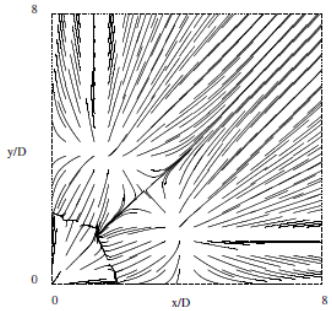


(b) pathlines at $0.01D$ above the impingement plane

Figure 4.6: Pathlines for the square set-up, as predicted by the $k-\varepsilon$ model. One can clearly observe the asymmetry in the predicted flow field. A vortex has developed near the top-left jet (see 4.6(b)); this jet is disturbed and deflected outwards (4.6(a)).



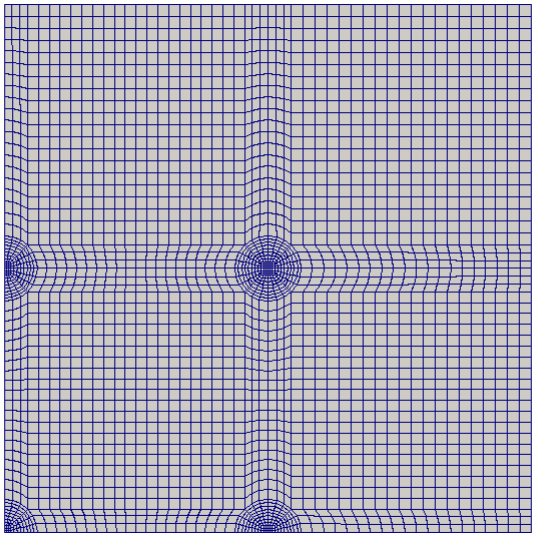
(a) the computational domain



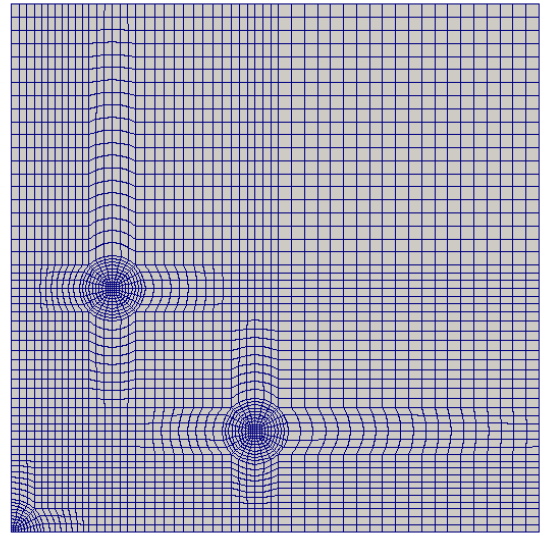
(b) pathlines at $0.01D$ above the impingement plane

e 4.14: Pathlines for the circular set-up, as predicted by the $k-\varepsilon$ model. symmetry in the flow field can be observed.

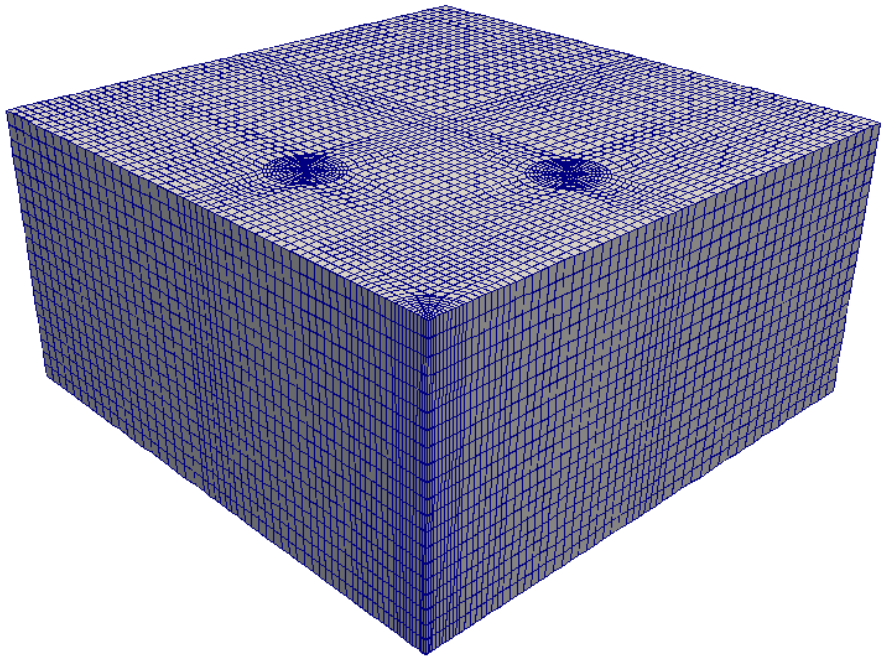
・Ercoftac SIG15 testcase　case11.4　・・・　1/4 mesh



square

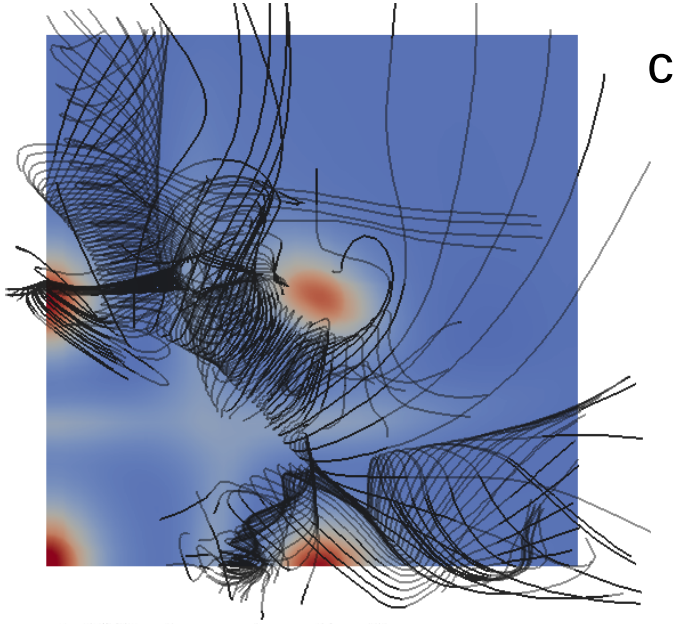


circular

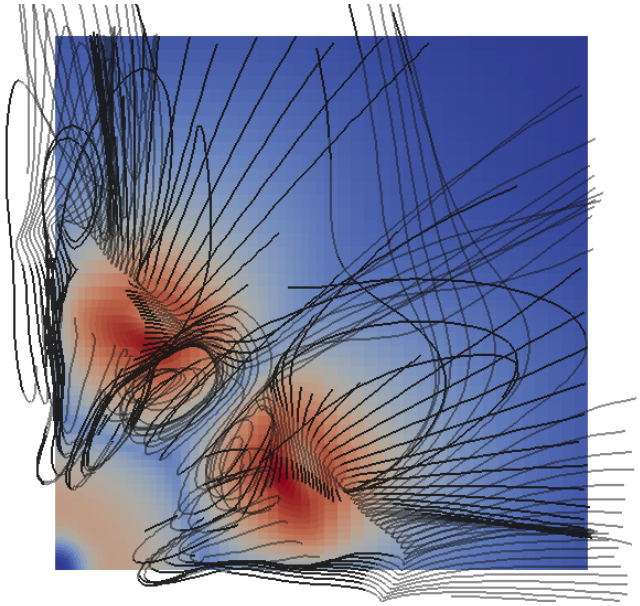


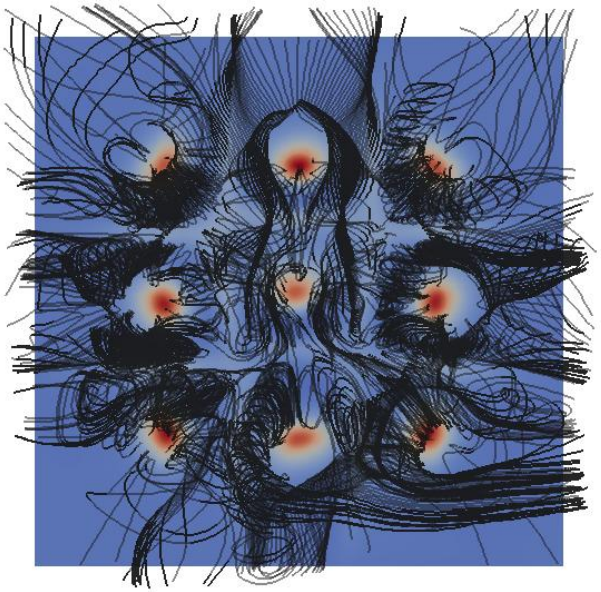
- Ercoftac SIG15 testcase case11.4

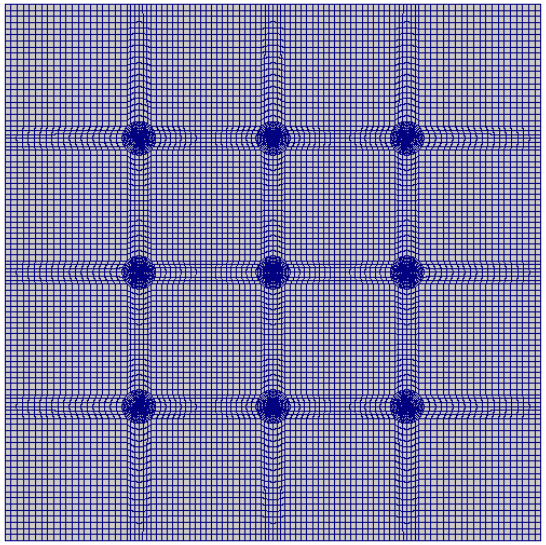
square



circular







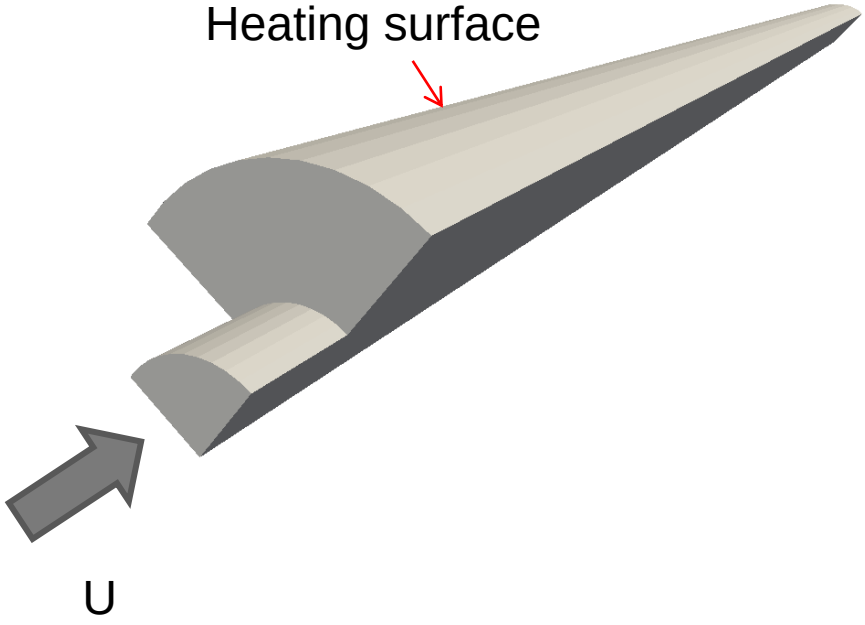
square Full model

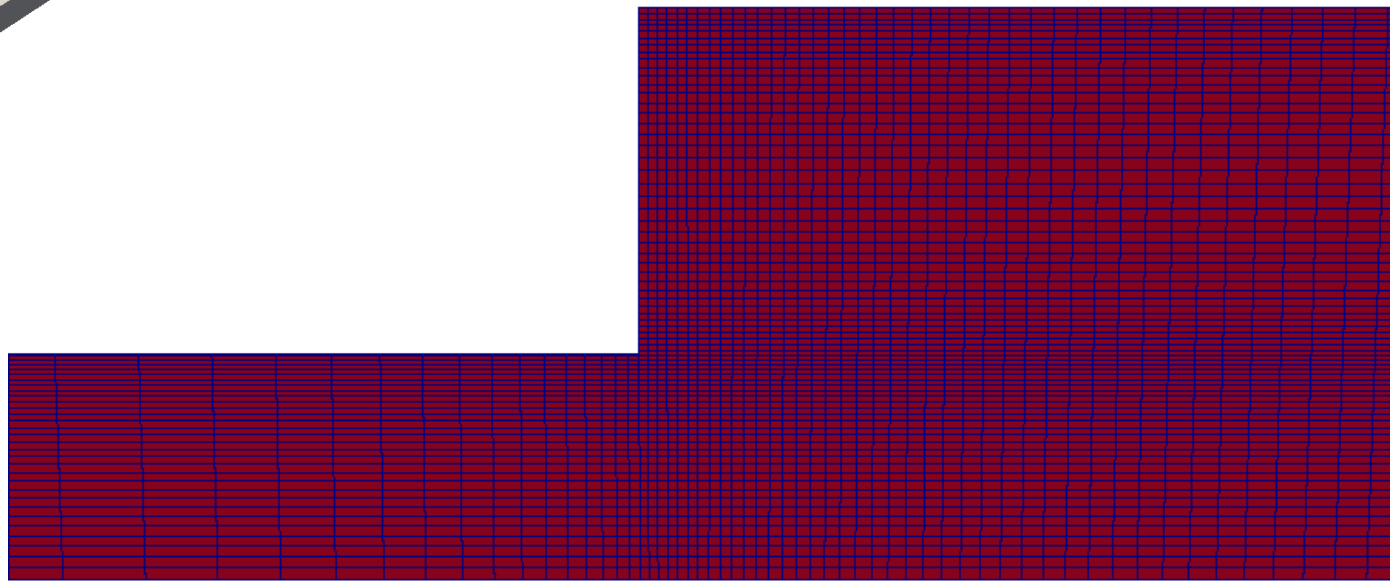
・Ercoftac SIG15 testcase

Case15.1 axisymmetric expansion pipe with heat transfer

Re≒40000、exp(Baughn et al 1984)

OF -> buoyantBoussinesqSimpleFoam





・Ercoftac SIG15 testcase　Tのsurface field・・・壁面にHeatFlux BC

```
    step
    {
//type zeroGradient;
type groovyBC;
value uniform 0;
gradientExpression "gradT";
fractionExpression "0";
variables "Cp0=1000;rho0=1.0;q=0.0;gradT=q/(kappaEff*Cp0*rho0);";
timelines ();

/**
type turbulentHeatFluxTemperature;
alphaEff kappaEff;
value uniform 0;
Cp Cp;
q uniform 0; // [W/m^2]
**/
    }
    heat
    {
type groovyBC;
value uniform 0;
gradientExpression "gradT";
fractionExpression "0";
variables "Cp0=1000;rho0=1.0;q=0.3;gradT=q/(kappaEff*Cp0*rho0);";
timelines ();

/**
type turbulentHeatFluxTemperature;
alphaEff kappaEff;
value uniform 0;
Cp Cp;
q uniform 0.3; // [W/m^2]
**/
    }
```

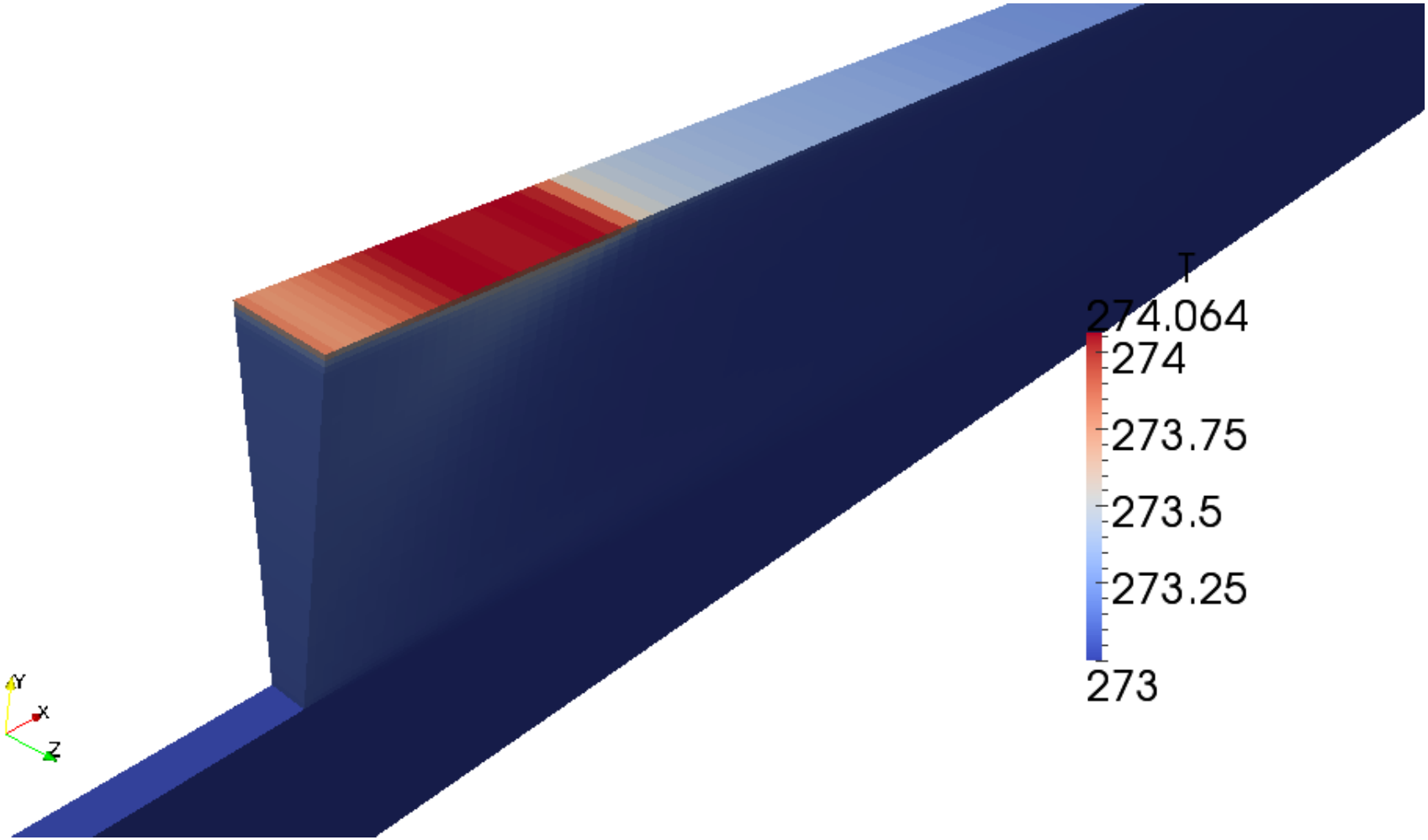
groovyBC

groovyBC

このHeatFluxはCFXのサンプルデータから引用

原文にはそれらしい記述がなかった・・・

・Ercoftac SIG15 testcase



はたしてこれで合っているのかどうか・・・？
Nu数を算出してみないとわからない

・乱流モデルについて

CFD On-line上で見つけた幾つかのモデルをOF 1.6-ext上で実行、違うバージョンでインプリメントされたものは多少修正して、サンプルモデル計算

①V2F　4方程式モデル(Durbin1995)
http://www.cfd-online.com/Forums/openfoam-solving/58440-v2f-turbulence-model.html

②低Re数型RSM
　Durbin Elliptic Relaxation Model (Durbin1993)
　Hanjalic Elliptic Blending Model (Manceau2002)　} **SSG(Speziale et al 、1991)**
http://www.cfd-online.com/Forums/openfoam-solving/95185-reynolds-stress-transport-model-elliptic-relaxation.html

③gamma-ReTheta遷移モデル(Langtry 2009)
http://www.cfd-online.com/Forums/openfoam-programming-development/85382-gamma-retheta-turbulence-model-predicting-transitional-flows.html

・・・＞CFD online上で作者(Felix.L)がv1.6、2.0に対応したソースを公開

④標準k-eモデルのLaunder-Kato版(LK1993)

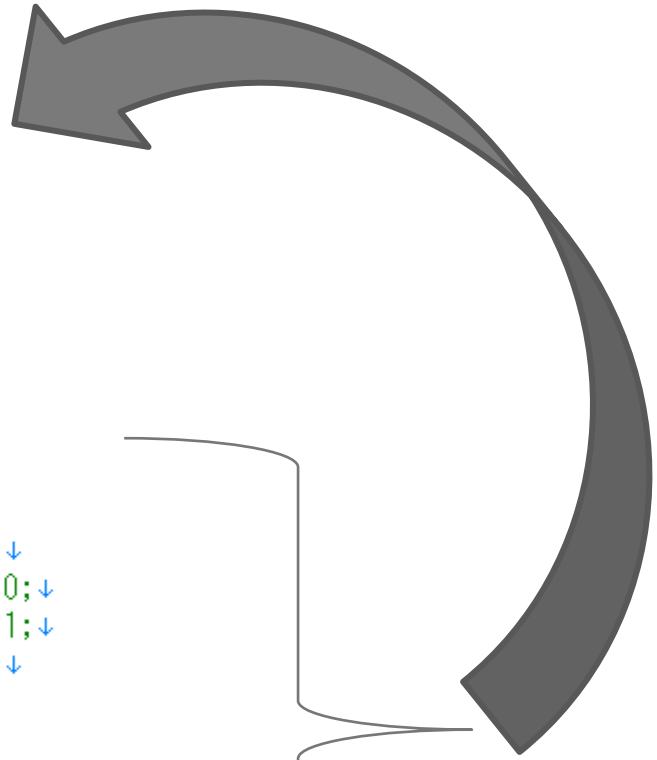
・乱流モデルについて　V2F Model

オリジナルはOF1.5なので、1.6-extで動くように一部修正

```
bool DurbinV2F::read()
{
    if (RASModel::read())
    {
        Cmu_.readIfPresent(coeffDict());
        CmuKE_.readIfPresent(coeffDict());
        Ceps10_.readIfPresent(coeffDict());
        Ceps11_.readIfPresent(coeffDict());
        Ceps2_.readIfPresent(coeffDict());
        C1_.readIfPresent(coeffDict());
        C2_.readIfPresent(coeffDict());
        CL_.readIfPresent(coeffDict());
        CEta_.readIfPresent(coeffDict());
        oneOnSigmaK_.readIfPresent(coeffDict());
        oneOnSigmaEps_.readIfPresent(coeffDict());
        yStarLim_.readIfPresent(coeffDict());

/**      before OF1.5 implement
        turbulenceModelCoeffs_.lookup("Cmu") >> Cmu;
        turbulenceModelCoeffs_.lookup("CmuKE") >> CmuKE;
        turbulenceModelCoeffs_.lookup("Ceps10") >> Ceps10;
        turbulenceModelCoeffs_.lookup("Ceps11") >> Ceps11;
        turbulenceModelCoeffs_.lookup("Ceps2") >> Ceps2;
        turbulenceModelCoeffs_.lookup("C1") >> C1;
        turbulenceModelCoeffs_.lookup("C2") >> C2;
        turbulenceModelCoeffs_.lookup("CL") >> CL;
        turbulenceModelCoeffs_.lookup("CEta") >> CEta;
        turbulenceModelCoeffs_.lookup("oneOnSigmaK") >> oneOnSigmaK;
        turbulenceModelCoeffs_.lookup("oneOnSigmaEps") >> oneOnSigmaEps;
        turbulenceModelCoeffs_.lookup("yStarLim") >> yStarLim;
**/

        return true;
```



・乱流モデルについて　V2F Model

```
    volScalarField S2 = 2*magSqr(symm(fvc::grad(U_)));

    volScalarField G = nut_*S2;

    volScalarField T_ = T();
    volScalarField Ceps1_= Ceps10_*(scalar(1.0)+Ceps11_*min(sqrt(k_/v2_),scalar(10.0)));
//    volScalarField Ceps1 = Ceps10*(scalar(1.0)+Ceps11*sqrt(k_/v2_));

    // Dissipation rate equation

#   include "epsilonV2FWallI.H"
    tmp<fvScalarMatrix> epsEqn
    (
        fvm::ddt(epsilon_)
      + fvm::div(phi_, epsilon_)
      - fvm::laplacian(DepsilonEff(), epsilon_)
     ==
        Ceps1_*G/T_
      - fvm::Sp(Ceps2_/T_, epsilon_)
    );

    epsEqn().relax();
#   include "wallDissipationV2FI.H"
    solve(epsEqn);
    bound(epsilon_, epsilon0_);


    // Turbulent kinetic energy equation

    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(k_)
      + fvm::div(phi_, k_)
      - fvm::laplacian(DkEff(), k_)
     ==
        G - fvm::Sp(1.0/T_, k_)
//        G - fvm::Sp(epsilon_/k_, k_)
    );

    kEqn().relax();
    solve(kEqn);
    bound(k_, k0_);
```

・乱流モデルについて　V2F Model

```
    // f equation
    volScalarField L_ = L();

    tmp<fvScalarMatrix> fEqn
    (
//      - fvm::laplacian(scalar(1.0),f_)
      - fvm::laplacian(f_)
     ==
      - fvm::Sp(1.0/sqr(L_),f_)
      - ((C1_-scalar(6.0))*v2_/k_ - 2.0/3.0*(C1_-scalar(1.0)))/(sqr(L_)*T_)
      + C2_*G/(k_*sqr(L_))
/*      - fvm::Sp(1.0/sqr(L_),f_)
      - ((C1-scalar(1.0))*(v2_/k_-scalar(2.0/3.0))/T_
       - C2*G/k_
       - 5.0*epsilon_*v2_/sqr(k_))/sqr(L_)*/
    );

    fEqn().relax();
    solve(fEqn);
    bound(f_, f0_);
```

```
    // v2 equation

    tmp<fvScalarMatrix> v2Eqn
    (
        fvm::ddt(v2_)
      + fvm::div(phi_, v2_)
      - fvm::laplacian(DkEff(), v2_)
     ==
//    k_*f_
// Davidson correction - 2003
        min(k_*f_,
          -((C1_-scalar(6.0))*v2_ - 2.0/3.0*k_*(C1_-scalar(1.0)))/T_ + C2_*G)
      - fvm::Sp(6.0*epsilon_/k_, v2_)
    );

    v2Eqn().relax();
    solve(v2Eqn);
    bound(v2_, k0_);

    // Re-calculate viscosity (with Davidson correction - 2003)
    nut_ = min(CmuKE_*sqr(k_)/(epsilon_ + epsilonSmall_),
               Cmu_*v2_*T());
```

・乱流モデルについて　Durbin Elliptic Relaxation Model → SSG
SSGモデル（RSMの圧力-ひずみ相関項が2次、 流線曲率のある流れ効果有）
Fluent13　Theory Manualから抜粋

$$\phi_{ij} = -(C_1\rho\varepsilon + C_1^* P)b_{ij} + C_2\rho\varepsilon\left(b_{ik}b_{kj} - \frac{1}{3}b_{mn}b_{mn}\delta_{ij}\right) + \left(C_3 - C_3^*\sqrt{b_{ij}b_{ij}}\right)\rho k S_{ij}$$

$$+ C_4\rho k\left(b_{ik}S_{jk} + b_{jk}S_{ik} - \frac{2}{3}b_{mn}S_{mn}\delta_{ij}\right) + C_5\rho k\left(b_{ik}\Omega_{jk} + b_{jk}\Omega_{ik}\right)$$

where b_{ij} is the Reynolds-stress anisotropy tensor defined as

$$b_{ij} = -\left(\frac{-\rho\overline{u_i'u_j'} + \frac{2}{3}\rho k\delta_{ij}}{2\rho k}\right)$$

The mean strain rate, S_{ij}, is defined as

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right)$$

The mean rate-of-rotation tensor, Ω_{ij}, is defined by

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right)$$

The constants are

$$C_1 = 3.4, C_1^* = 1.8, C_2 = 4.2, C_3 = 0.8, C_3^* = 1.3, C_4 = 1.25, C_5 = 0.4$$

```
    volSymmTensorField bij("bij", 0.5*dev(R_/k_));
    volTensorField gradU = fvc::grad(U_);
    volSymmTensorField Sij("Sij", dev(symm(gradU)));
    volTensorField Wij = skew(gradU);

tmp<fvSymmTensorMatrix> REqn
    (
        fvm::ddt(R_)
        + fvm::div(phi_, R_)
        + fvm::SuSp(-fvc::div(phi_), R_)
      //- fvm::laplacian(Cs*(k_/epsilon_)*R_, R_)
        - fvm::laplacian(DREff(), R_)
        + fvm::Sp(epsilon_/k_, R_)
        ==
        P                                          // production tensor
        - (Cg1_*epsilon_ + Cg1s_*G)*bij
        + Cg2_*epsilon_*dev(symm(bij & bij))
        + (Cg3_ - Cg3s_*sqrt(bij && bij))*k_*Sij
        + Cg4_*k_*dev(twoSymm(symm2full(bij) & Sij))
        + Cg5_*k_*(twoSymm(bij & Wij))
    );
```

OFのデフォルトにはない関数

・乱流モデルについて　sym2full

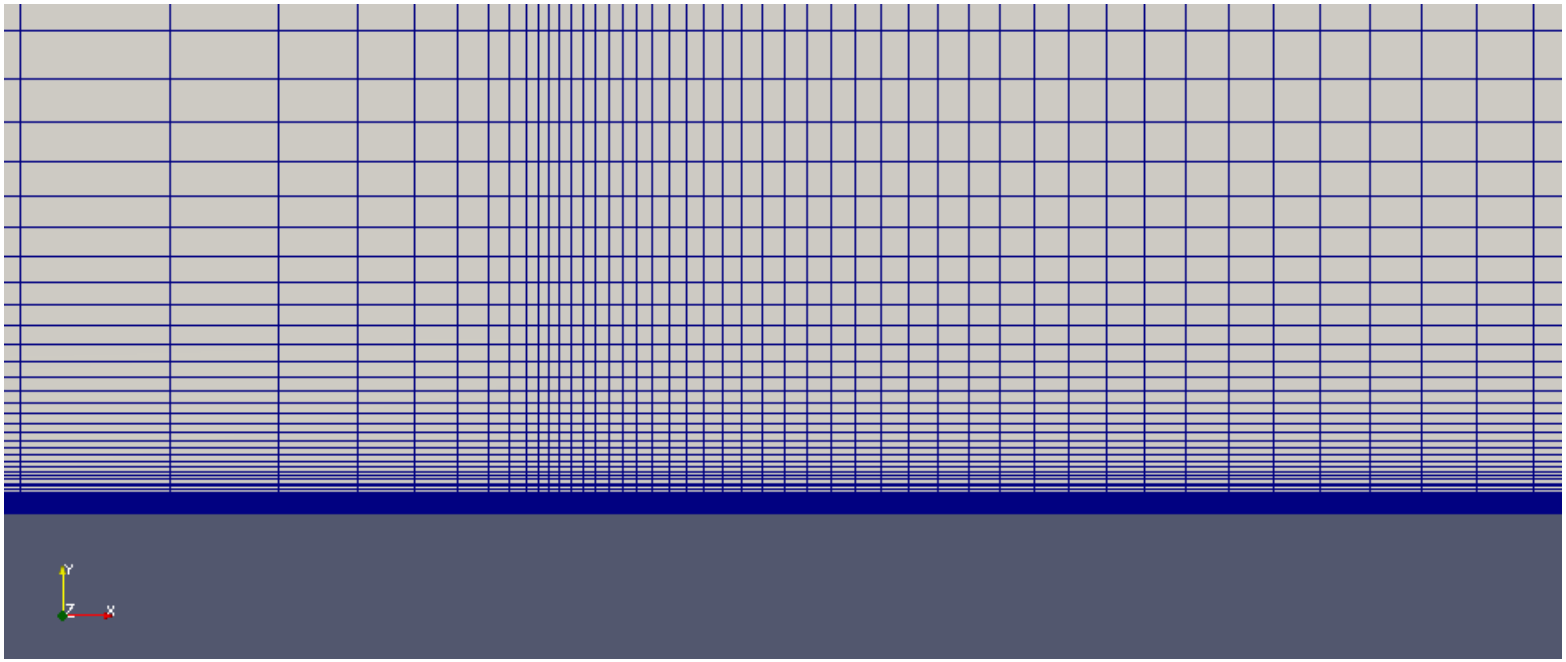
```
typedef GeometricField<Tensor<Cmpt>, PatchField, GeoMesh> FieldType;
tmp<FieldType> tFull
(
    new FieldType
    (
        IOobject
        (
            symm.name() + "FullTensor",
            symm.mesh().time().timeName(),
            symm.mesh(),
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        symm.mesh(),
        dimensionedTensor("0", symm.dimensions(), Tensor<Cmpt>::zero),
        symm.boundaryField().types()
    )
);

FieldType& full = tFull();

// manipulate components
full.replace(Tensor<Cmpt>::XX, symm.component(SymmTensor<Cmpt>::XX));
full.replace(Tensor<Cmpt>::YY, symm.component(SymmTensor<Cmpt>::YY));
full.replace(Tensor<Cmpt>::ZZ, symm.component(SymmTensor<Cmpt>::ZZ));
full.replace(Tensor<Cmpt>::XY, symm.component(SymmTensor<Cmpt>::XY));
full.replace(Tensor<Cmpt>::YZ, symm.component(SymmTensor<Cmpt>::YZ));
full.replace(Tensor<Cmpt>::XZ, symm.component(SymmTensor<Cmpt>::XZ));
full.replace(Tensor<Cmpt>::ZX, symm.component(SymmTensor<Cmpt>::XZ));
full.replace(Tensor<Cmpt>::YX, symm.component(SymmTensor<Cmpt>::XY));
full.replace(Tensor<Cmpt>::ZY, symm.component(SymmTensor<Cmpt>::YZ));

return tFull;
```

・乱流モデルについて　gamma-ReTheta遷移モデル（sample - T3A test case)



・乱流モデルについて　Launder-Kato k-ε / MMK

```
// MMK(Murakami, Mochida, Kondoh) model implement -> modified LK
    volTensorField gradU = fvc::grad(U_);
    volScalarField S2 = 2*magSqr(dev(symm(gradU)));
    volScalarField magS = sqrt(S2);

    volScalarField V2 = 2*magSqr(dev(skew(gradU)));
    volScalarField magV = sqrt(V2);

    // Re-calculate viscosity
//    volScalarField threshold = magV / magS;

//    if(threshold > 1.0)
    if(magV > magS )
    {
    nut_ = Cmu_*sqr(k_)/(epsilon_ + epsilonSmall_);
    }
    else
    {
    nut_ = Cmu_*magV/magS*sqr(k_)/(epsilon_ + epsilonSmall_);
    }

    nut_.correctBoundaryConditions();
}
```

Fluent13　Theory Manualから抜粋

4.3.4. Modeling Turbulent Production in the k-ε Models

The term G_k, representing the production of turbulence kinetic energy, is modeled identically for the standard, RNG, and realizable k-ε models. From the exact equation for the transport of k, this term may be defined as

$$G_k = -\rho \overline{u_i' u_j'} \frac{\partial u_j}{\partial x_i} \qquad (4\text{–}54)$$

To evaluate G_k in a manner consistent with the Boussinesq hypothesis,

$$G_k = \mu_t S^2 \qquad (4\text{–}55)$$

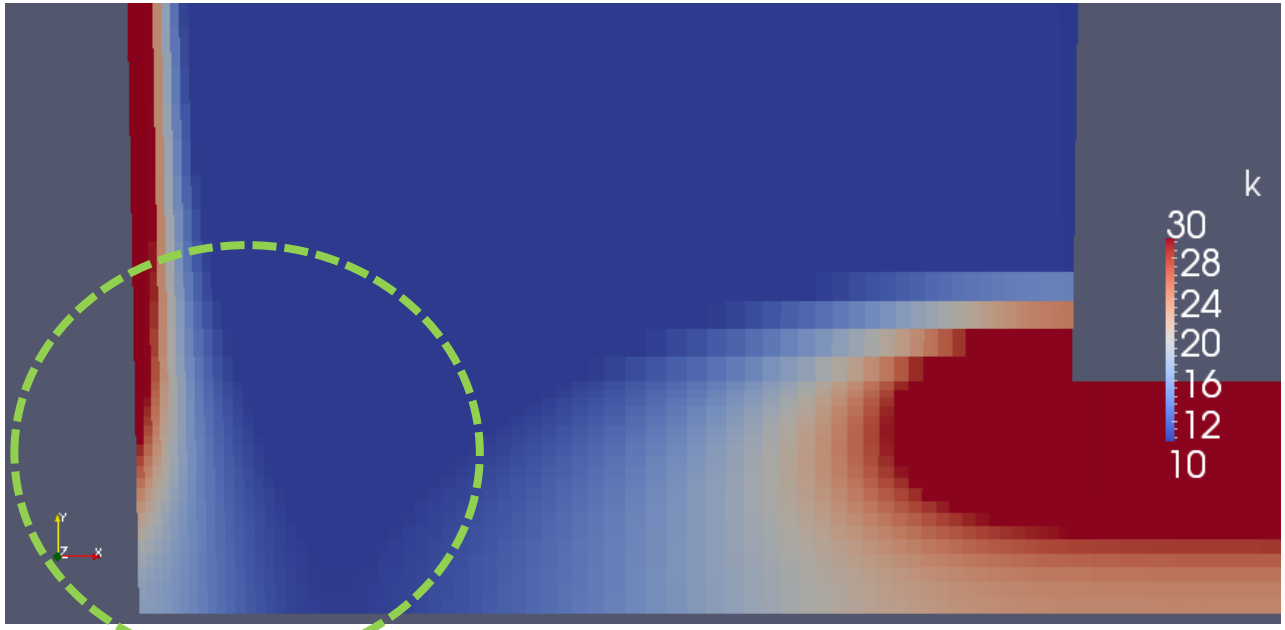
where S is the modulus of the mean rate-of-strain tensor, defined as

$$S \equiv \sqrt{2 S_{ij} S_{ij}} \qquad (4\text{–}56)$$

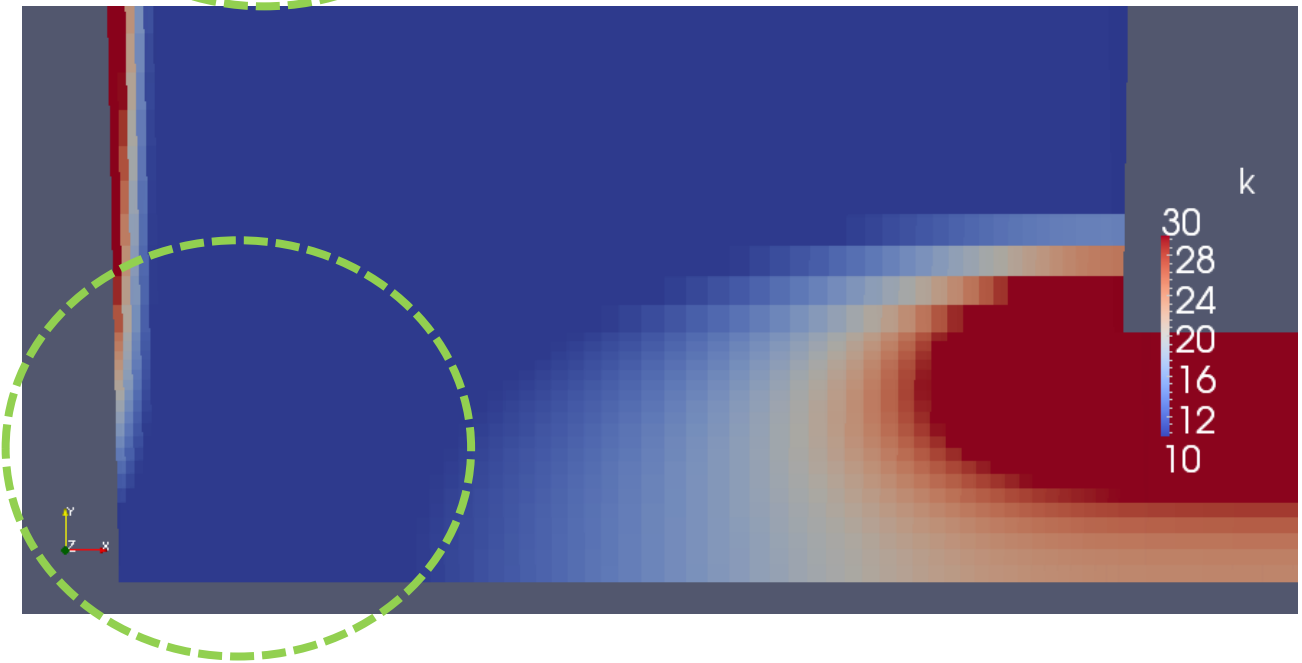
Important

When using the high-Reynolds number k-ε versions, μ_{eff} is used in lieu of μ_t in *Equation 4–55* (p. 57).

・乱流モデルについて　turbulent kinematic energy



Standard k-ε



Launder-Kato k-ε

・・・衝突領域の過大な乱流運動エネルギー生成を抑制