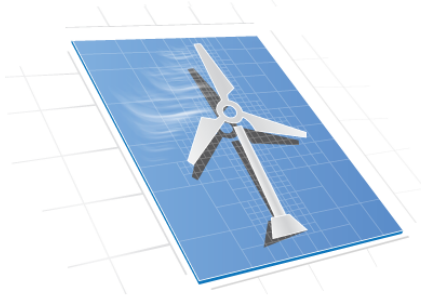


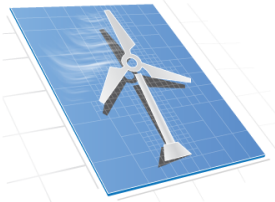
HyperCFD
Fertınaz Yazılım Ltd.

# OpenFOAM Workshop
# Teknopark Istanbul

## 04 IC, BC & Turbulence
## January 2017

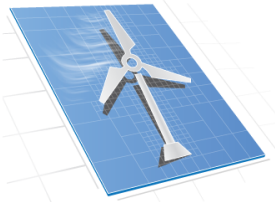
# HyperCFD – Disclaimer

This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OpenFOAM and OpenCFD trademarks.



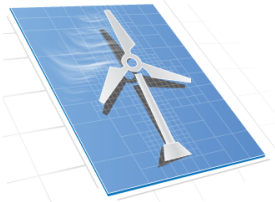
# HyperCFD – OpenFOAM IC's and BC's

- The idea of `OpenFOAM` is to solve fluid flow problem either on internal or external meshes which means solving an Initial Boundary Value Problem (IVBP)
- Therefore we need to define Initial and Boundary conditions
- Initial conditions define the initial state
- Boundary conditions close the problem
- They indeed have to be physically realistic
- They have to be defined for all variables that are solved
  - For an incompressible laminar problem problem they are **U** and p



# HyperCFD – OpenFOAM Boundary Conditions

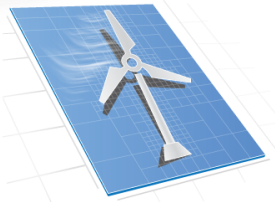
- BC's can be
  - Dirichlet: Value of a variable prescribed at the boundary – Fixed value
  - Neumann: Grandient normal at the boundary is prescribed – Fixed gradient
  - Robin: A mixture of both
- Detect the right patch and right type
- Avoid solid regions in the vicinity of the boundaries
- Be careful with the treatment of the backflow either at the inlet or outlet



# HyperCFD – OpenFOAM Boundary Conditions

- There are two types of boundary conditions in OpenFOAM
  - Base type: Boundary patches defined in the `blockMeshDict` and `polyMesh/boundary`
  - If you use a converted mesh generated by a different software, you might want to play
    with this file for consistency
  - Names are generic
  - Types must be consistent

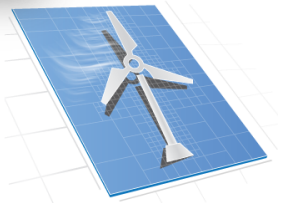
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

3
(
    movingWall
    {
        type            wall;
        nFaces          20;
        startFace       760;
    }
    fixedWalls
    {
        type            wall;
        nFaces          60;
        startFace       780;
    }
    frontAndBack
    {
        type            empty;
        nFaces          800;
        startFace       840;
    }
)

// ************************************************************************* //
```



# HyperCFD – OpenFOAM Boundary Conditions

- There are two types of boundary conditions in OpenFOAM
  - Primitive type: Boundary conditions defined in the field variables files `0/U, 0/p`
  - These types must be consistent with base types
  - For a `patch` primitive types are
    - `fixedValue`
    - `zeroGradient`
    - `inletOutlet`
    - `outletInlet`
    - and many more …
  - For a `wall` primitive types are
    - `fixedValue`
    - `zeroGradient`

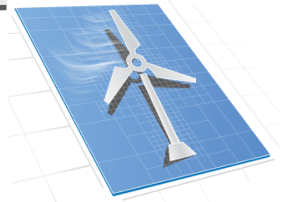
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

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

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```



# HyperCFD – OpenFOAM Boundary Conditions

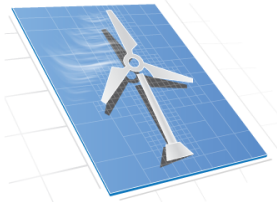
- Boundary conditions for a typical case

| Boundary Type | Velocity | Pressure | Turbulence fields |
|---|---|---|---|
| Inlet | fixedValue | zeroGradient | fixedValue |
| Outlet | inletOutlet | fixedValue | inletOutlet |
| Wall | fixedValue | zeroGradient | wallfunctions |
| Symmetry | symmetry | symmetry | symmetry |
| Periodic | cyclic | cyclic | cyclic |
| Empty | empty | empty | empty |
| Slip wall | slip | slip | slip |



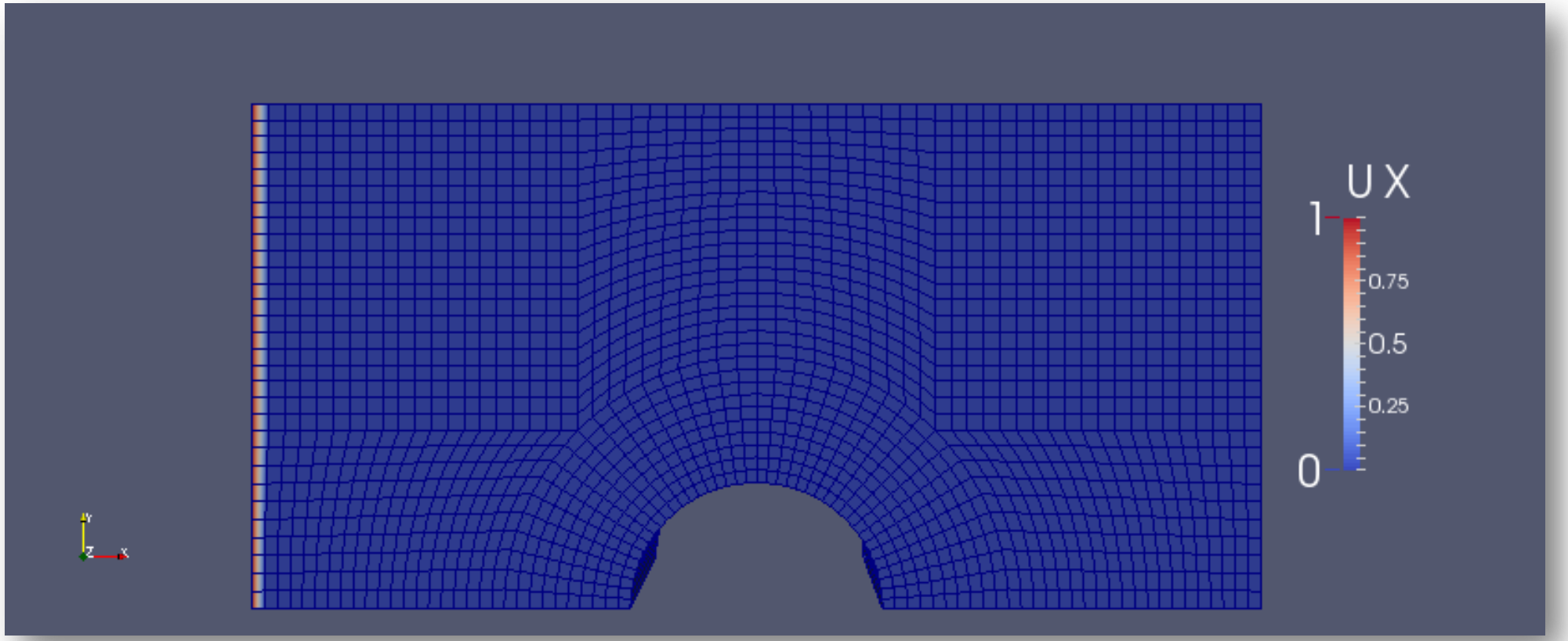
# HyperCFD – OpenFOAM Initial Conditions

- Initial conditions define the initial state
- A good IC can lead quicker convergence
- Initial conditions in OpenFOAM can be
  - Uniform
  - Non-uniform: These can be extracted from other simulations and math functions
- Detect the right patch and right type



# HyperCFD – OpenFOAM Initial Conditions

- Lets see a sample case, laminar and incompressible

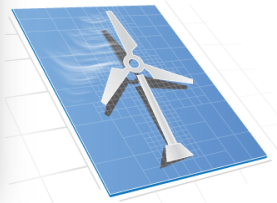


# HyperCFD – OpenFOAM Initial Conditions

- Lets see a sample case, laminar and incompressible
  - `0/U`
  - `0/p`

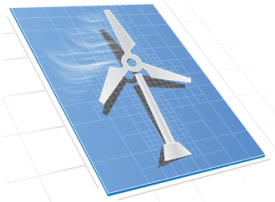
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    down
    {
        type            symmetryPlane;
    }

    right
    {
        type            zeroGradient;
    }

    up
    {
        type            symmetryPlane;
    }

    left
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    cylinder
    {
        type            symmetryPlane;
    }

    defaultFaces
    {
        type            empty;
    }
}

// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    down
    {
        type            symmetryPlane;
    }

    right
    {
        type            fixedValue;
        value           uniform 0;
    }

    up
    {
        type            symmetryPlane;
    }

    left
    {
        type            zeroGradient;
    }

    cylinder
    {
        type            symmetryPlane;
    }

    defaultFaces
    {
        type            empty;
    }
}

// ************************************************************************* //
```



# HyperCFD – OpenFOAM Turbulence

- We cannot discuss theory behind the turbulence modelling
- A simple overview:
  - Random and chaotic
  - No general model that fits for all
  - Lots of different models, all have advantages and disadvantages
    - RANS/URANS: Time averaged NS with turbulence models, appropriate for many applications
    - DES/LES: Solve filtered NS, computationally expensive
    - DNS: Computationally very expensive

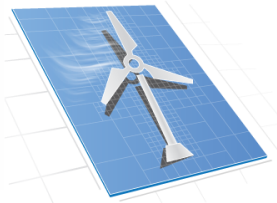


# HyperCFD – OpenFOAM Turbulence

- How to enable turbulence:
  `constant/turbulenceProperties`

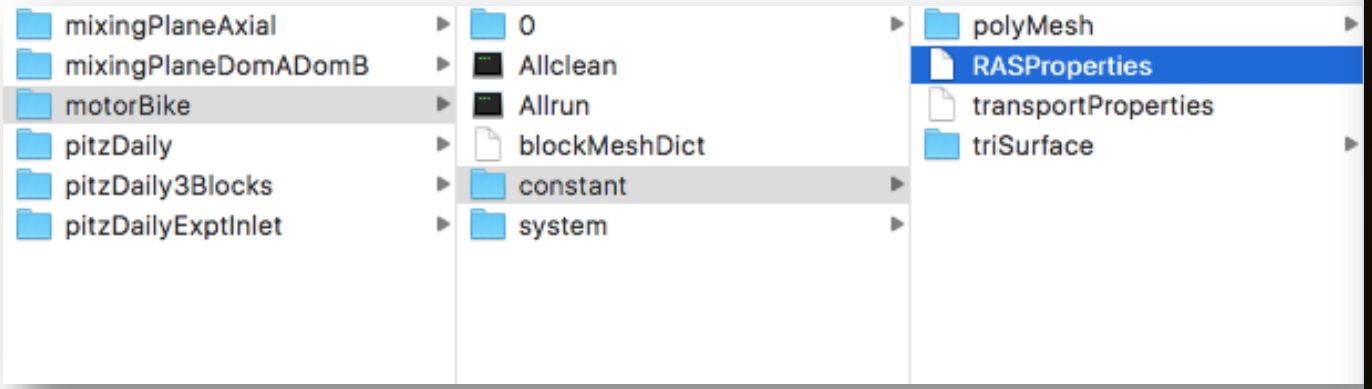
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RAS;

RAS
{
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType RAS;

RAS
{
    RASModel            kEpsilon;

    turbulence          on;

    printCoeffs         on;

    kEpsilonCoeffs
    {
        Cmu         0.09;
        C1          1.44;
        C2          1.92;
        sigmaEps    1.11; //Original value:1.44
        // See:
        // D.M. Hargreaves and N.G. Wright
        // "On the use of the k-Epsilon model in commercial CFD software
        // to model the neutral atmospheric boundary layer",
        // J. of wind engineering and industrial aerodynamics,
        // 95(2007) 355-269
    }
}

// ************************************************************************* //
```

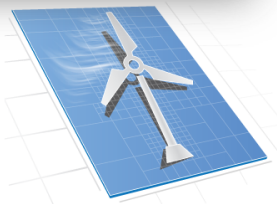


# HyperCFD – OpenFOAM Turbulence

- How to enable turbulence: Older versions
  `constant/RASProperties`

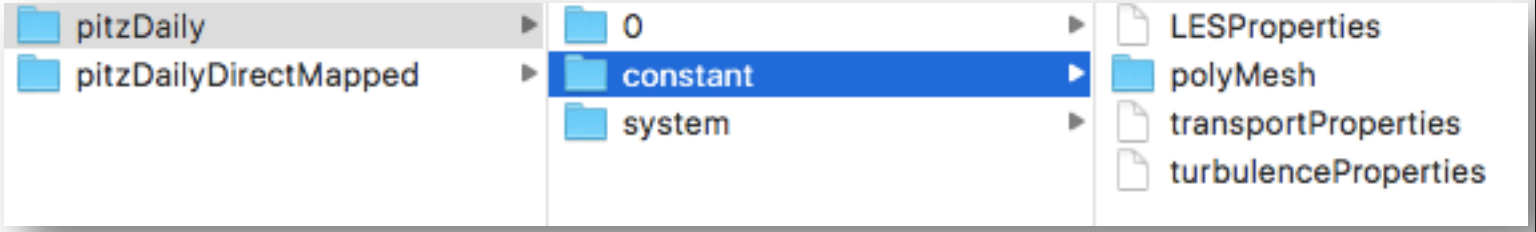



```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | foam-extend: Open Source CFD                    |
|  \\    /   O peration     | Version:     3.1                                |
|   \\  /    A nd           | Web:         http://www.extend-project.de       |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel            kOmegaSST;

turbulence          on;

printCoeffs         on;

// ************************************************************************* //
```

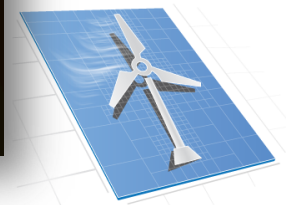


# HyperCFD – OpenFOAM Turbulence

- How to enable turbulence: Older versions
  `constant/LESProperties`




```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | foam-extend: Open Source CFD                    |
|  \\    /   O peration     | Version:     3.1                                |
|   \\  /    A nd           | Web:         http://www.extend-p
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      LESProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

LESModel        oneEqEddy;

delta           cubeRootVol;

printCoeffs     on;

cubeRootVolCoeffs
{
    deltaCoeff      1;
}

PrandtlCoeffs
{
    delta           cubeRootVol;
    cubeRootVolCoeffs
    {
        deltaCoeff      1;
    }
```

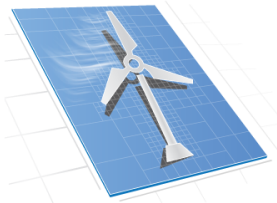


# HyperCFD – OpenFOAM Turbulence

- List of available turbulence models in OpenFOAM
- You can also type `foamList -incompressibleTurbulenceModels`

| **RAS turbulence models** — *RASModels* | |
|---|---|
| *laminar* | Dummy turbulence model for laminar flow |
| *kEpsilon* | Standard $k-\varepsilon$ model |
| *kOmega* | $k-\omega$ model |
| *kOmegaSST* | $k-\omega-SST$ model |
| *kOmegaSSTLM* | Langtry-Menter 4-equation transitional SST model |
| *kOmegaSSTSAS* | $k-\omega-SST-SAS$ model |
| *LaunderSharmaKE* | Launder-Sharma low-$Re$ $k-\varepsilon$ model |
| *LRR* | Launder-Reece-Rodi RSTM |
| *realizableKE* | Realizable $k-\varepsilon$ model |
| *RNGkEpsilon* | $RNG-k-\varepsilon$ model |
| *SpalartAllmaras* | Spalart-Allmaras 1-eqn mixing-length model |
| *SSG* | Speziale, Sarkar and Gatski Reynolds-stress model |
| *v2f* | $v2-f$ model |

Taken from http://www.openfoam.com/documentation/user-guide/standard-libraries.php Table A.5

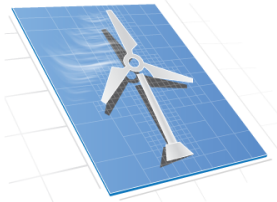


# HyperCFD – OpenFOAM Turbulence

- List of available turbulence models in OpenFOAM

| **Large-eddy simulation (LES) filters** — *LESfilters* | |
|---|---|
| *laplaceFilter* | Laplace filters |
| *simpleFilter* | Simple filter |
| *anisotropicFilter* | Anisotropic filter |
| **Large-eddy simulation deltas** — *LESdeltas* | |
| *PrandtlDelta* | Prandtl delta |
| *cubeRootVolDelta* | Cube root of cell volume delta |
| *maxDeltaxyz* | Maximum of x, y and z; for structured hex cells only |
| *smoothDelta* | Smoothing of delta |
| **LES turbulence models** — *LESModels* | |
| *DeardorffDiffStress* | Differential SGS Stress model |
| *dynamicKEqn* | Dynamic one equation eddy-viscosity |
| *dynamicLagrangian* | Dynamic SGS model with Lagrangian averaging |
| *kEqn* | One equation eddy-viscosity model |
| *Smagorinsky* | Smagorinsky SGS model |
| *WALE* | Wall-adapting local eddy-viscosity (WALE) model |
| **DES turbulence models** — *DESModels* | |
| *kOmegaSSTDES* | $k-omega-SST$ delayed eddy simulation (DES) model |
| *kOmegaSSTDDES* | $k-omega-SST$ delayed detached eddy simulation (DDES) model |
| *kOmegaSSTIDDES* | $k-omega-SST$ improved delayed detached eddy simulation (DDES) model |
| *SpalartAllmarasDES* | Spalart-Allmaras delayed eddy simulation (DES) model |
| *SpalartAllmarasDDES* | Spalart-Allmaras delayed detached eddy simulation (DDES) model |
| *SpalartAllmarasIDDES* | Spalart-Allmaras improved delayed detached eddy simulation (DDES) model |

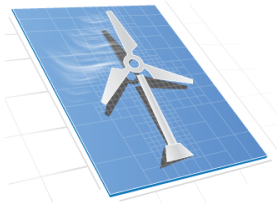
Taken from http://www.openfoam.com/documentation/user-guide/standard-libraries.php Table A.5



# HyperCFD – OpenFOAM Turbulence

- Wallfunctions in OpenFOAM
- Theory not in details:
  - Near wall treatment is crucial
  - One can resolve viscous sublayer when Re is low
  - For high Re you model it using wallfunctions
  - You can check what models are implemented in OpenFOAM library:
  - `$WM_PROJECT_DIR/src/TurbulenceModels/turbulenceModels/derivedFvPatchFields/wallFunctions`
  - Depending on the turbulence model you use there are different wallfunction

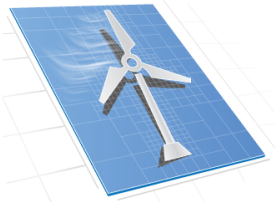
```
fertinaz fertinaz 4096 Maz 25  2016 epsilonWallFunctions
fertinaz fertinaz 4096 Maz 25  2016 fWallFunctions
fertinaz fertinaz 4096 Maz 25  2016 kqRWallFunctions
fertinaz fertinaz 4096 Maz 25  2016 nutWallFunctions
fertinaz fertinaz 4096 Maz 25  2016 omegaWallFunctions
fertinaz fertinaz 4096 Maz 25  2016 v2WallFunctions
```



# HyperCFD – OpenFOAM Turbulence

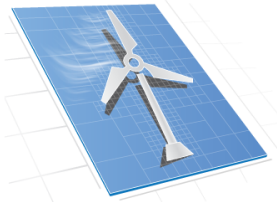
- Depending on the turbulence model you use, different quantities should be declared
- For one/two equation RANS models
  - k = Turbulent kinetic energy
  - epsilon = Turbulent dissipation
  - omega = Specific turbulent dissipation
  - nuTilda = Turbulent kinematic viscosity
  - nut = Turbulent viscosity
  - R = Reynolds stress tensor

```
Fatihs-MacBook-Air:ras fertinaz$ cd cavity/
Fatihs-MacBook-Air:cavity fertinaz$ ls
0               constant        system
Fatihs-MacBook-Air:cavity fertinaz$ cd 0
Fatihs-MacBook-Air:0 fertinaz$ ls -l
total 56
-rw-r--r--  1 fertinaz  staff  1223 Jun 17  2014 R
-rw-r--r--  1 fertinaz  staff  1187 Jun 17  2014 U
-rw-r--r--  1 fertinaz  staff  1233 Jun 17  2014 epsilon
-rw-r--r--  1 fertinaz  staff  1216 Jun 17  2014 k
-rw-r--r--  1 fertinaz  staff  1109 Jun 17  2014 nuTilda
-rw-r--r--  1 fertinaz  staff  1200 Jun 17  2014 nut
-rw-r--r--  1 fertinaz  staff  1103 Jun 17  2014 p
Fatihs-MacBook-Air:0 fertinaz$
```



# HyperCFD – OpenFOAM Field Initialization

- One should select the appropriate turbulence model  for the problem to be solved
- Decide whether to use wallfunctions or not
- Determine the fields need to declared
  - U, p, k, ω
  - U, p, k, ε
  - U, p, T etc.
- Then comes field initialization
  - $k = 3/2\ (\mathbf{U}\ \boldsymbol{I})^2$ where $\boldsymbol{I}$ is turbulent intensity
  - $\omega = (\rho\ k / \mu)\ (\mu_t / \mu)^{-1}$
  - $\mu$ is molecular dynamic viscosity and $\mu_t$ is turbulent eddy viscosity

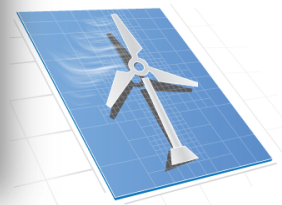


# HyperCFD – OpenFOAM Field Initialization

- Sample initial values of `k` and `epsilon` for a wind flow simulation

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

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

internalField   uniform $turbulentKE;

boundaryField
{
    outlet
    {
        type            inletOutlet;
        inletValue      uniform $turbulentKE;
        value           $internalField;
    }

    inlet
    {
        type            atmBoundaryLayerInletK;
        #include        "include/ABLConditions"
    }

    terrain
    {
        type            kqRWallFunction;
        value           uniform 0.0;
    }

    ground
    {
        type            zeroGradient;
    }

    #include "include/sideAndTopPatches"
}


// ************************************************************************* //
```

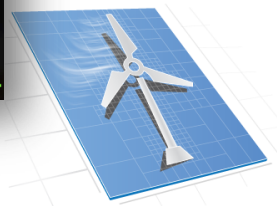
```
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

#include        "include/initialConditions"

internalField   uniform $turbulentEpsilon;

boundaryField
{
    terrain
    {
        type            epsilonWallFunction;
        Cmu             0.09;
        kappa           0.4;
        E               9.8;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      uniform $turbulentEpsilon;
        value           $internalField;
    }

    inlet
    {
        type            atmBoundaryLayerInletEpsilon;
        #include        "include/ABLConditions"
    }

    ground
    {
        type            zeroGradient;
    }

    #include "include/sideAndTopPatches"
}


// ************************************************************************* //
```



# HyperCFD – OpenFOAM Field Initialization

- And the `nut` file
  - `nut` is turbulent viscosity
  - OpenFOAM generates it but still one needs to be careful
  - This example uses a wallfunction
  - Choosing the right wallfunction completely depends on the physics of your problem

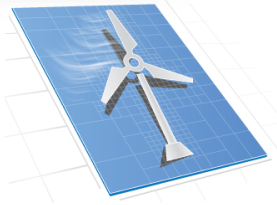
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    #include "include/ABLConditions"

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    terrain
    {
        type            nutkAtmRoughWallFunction;
        z0              $z0;
        value           uniform 0.0;
    }

    ground
    {
        type            calculated;
        value           uniform 0;
    }

    #include "include/sideAndTopPatches"
}


// ************************************************************************* //
```



# HyperCFD – OpenFOAM Field Initialization

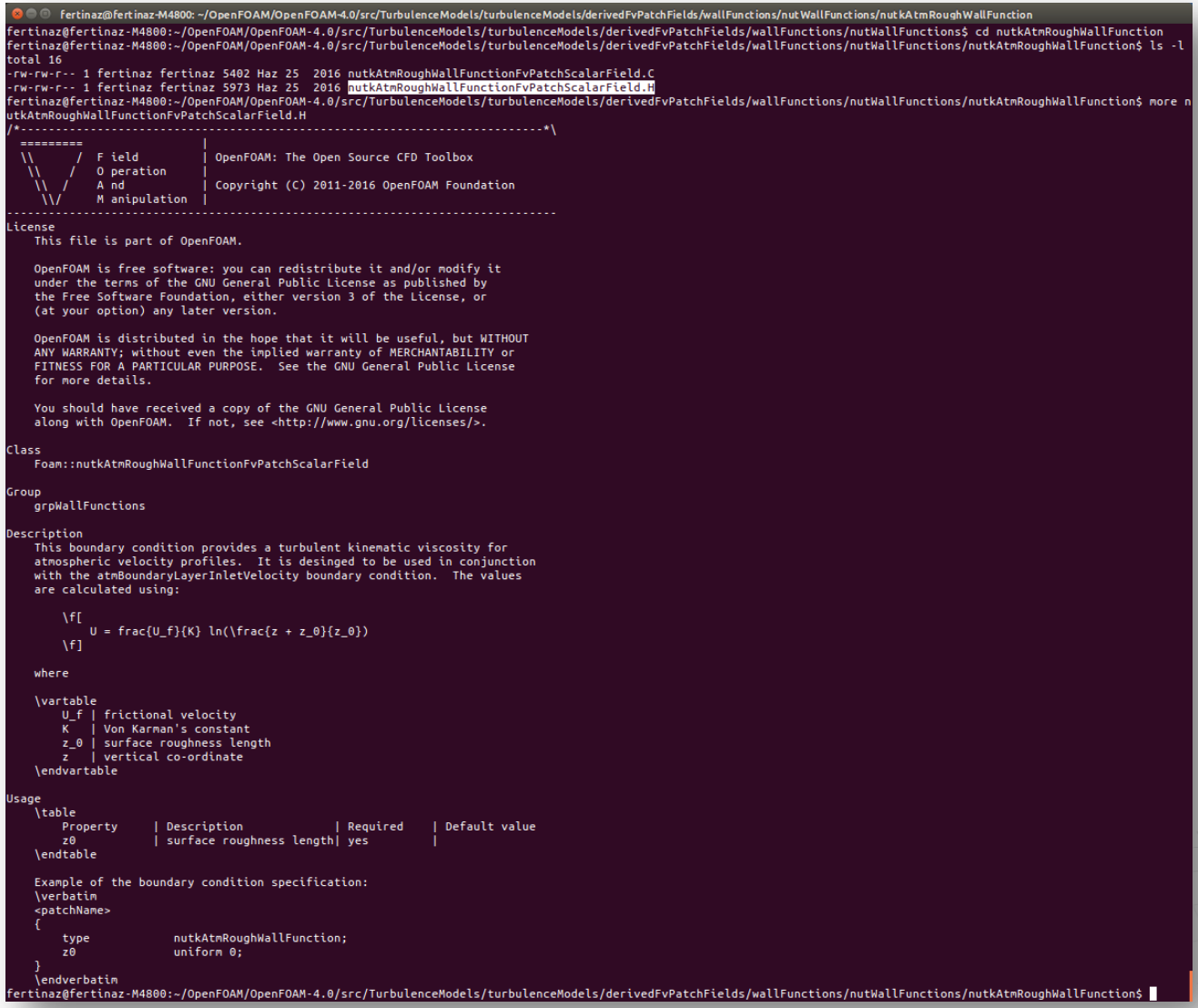
- Other `nut` related wallfunctions are:
- could also use something similar to `find $FOAM_SRC -name "nut*" -type d`

```
e/processor0/0$ src
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ ls
Allwmake          fileFormats             mesh             regionCoupled          thermophysicalModels
combustionModels  finiteVolume            meshTools        regionModels           topoChangerFvMesh
conversion        functionObjects         ODE              renumber               transportModels
dummyThirdParty   fvAgglomerationMethods  OpenFOAM         rigidBodyDynamics      triSurface
dynamicFvMesh     fvMotionSolver          OSspecific       rigidBodyMeshMotion    TurbulenceModels
dynamicMesh       fvOptions               parallel         sampling
edgeMesh          genericPatchFields      Pstream          sixDoFRigidBodyMotion
engine            lagrangian              randomProcesses  surfMesh
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ cd TurbulenceModels/
compressible/       incompressible/     phaseCompressible/   phaseIncompressible/ turbulenceModels/
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ cd TurbulenceModels/turbulenceModels/
Base/                 laminar/               lnInclude/             RAS/
derivedFvPatchFields/ LES/                   Make/                  ReynoldsStress/
eddyViscosity/        linearViscousStress/   nonlinearEddyViscosity/ TurbulenceModel/
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ cd TurbulenceModels/turbulenceModels/derivedFvPatchFields/
fixedShearStress/     porousBafflePressure/ wallFunctions/
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ cd TurbulenceModels/turbulenceModels/derivedFvPatchFields/
fixedShearStress/     porousBafflePressure/ wallFunctions/
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ cd TurbulenceModels/turbulenceModels/derivedFvPatchFields/wallFunct
ions/
epsilonWallFunctions/ kqRWallFunctions/     omegaWallFunctions/
fWallFunctions/       nutWallFunctions/     v2WallFunctions/
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src$ cd TurbulenceModels/turbulenceModels/derivedFvPatchFields/wallFunct
ions/nutWallFunctions/
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src/TurbulenceModels/turbulenceModels/derivedFvPatchFields/wallFunctions
/nutWallFunctions$ ls -l
total 36
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutkAtmRoughWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutkRoughWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutkWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutLowReWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutURoughWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutUSpaldingWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutUTabulatedWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutUWallFunction
drwxrwxr-x 2 fertinaz fertinaz 4096 Haz 25  2016 nutWallFunction
fertinaz@fertinaz-M4800:~/OpenFOAM/OpenFOAM-4.0/src/TurbulenceModels/turbulenceModels/derivedFvPatchFields/wallFunctions
/nutWallFunctions$
```



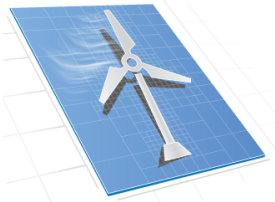
# HyperCFD – OpenFOAM Field Initialization

- Lets see what `nutAtmRoughWallFunction` is:
- Is it very important to know the contents at the moment?
  - No!
- But it is important to know where to find the source code
- And the description of each function



# HyperCFD – OpenFOAM Turbulence Remarks

- Closing remarks:
  - Select the most suitable model for your case: Wall bounded or free shear
  - If near wall treatment is not of your interest, use wallfunctions
  - Use $y^+$ to understand the grid resolution around the boundary layer
  - Know the **Re** of your problem, it provides information about turbulence and computational complexity
  - Initial and boundary conditions are important, choose carefully
  - Running `potentialFoam` to generate initial conditions is a well known technique



# HyperCFD – OpenFOAM Turbulence References

- Some recommended references on turbulence:
  - *Turbulent Flows*, Pope
  - *Turbulence for CFD*, Wilcox
  - *Fluid mechanics, turbulent flow and turbulence modelling*, Davidson