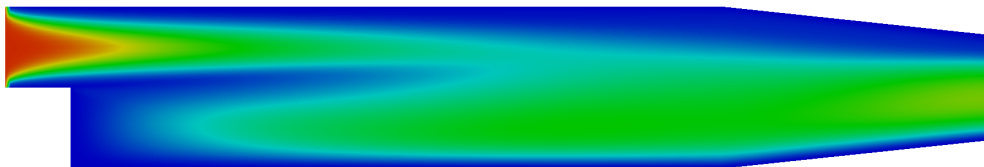


Tutorial Six

Turbulence – Steady State



5th edition, Sep. 2019



This offering is not approved or endorsed by ESI® Group, ESI-OpenCFD® or the OpenFOAM® Foundation, the producer of the OpenFOAM® software and owner of the OpenFOAM® trademark.

Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Editorial board:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek

Compatibility:

- OpenFOAM® 7
- OpenFOAM® v1906

Contributors:

- Bahram Haddadi
- Clemens Gößnitzer
- Jozsef Nagy
- Vikram Natarajan
- Sylvia Zibuschka
- Yitong Chen

Cover picture from:

- Bahram Haddadi

 Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions:

- Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- Noncommercial — You may not use this work for commercial purposes.
- Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

With the understanding that:

- Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.
- Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- Other Rights — In no way are any of the following rights affected by the license:
 - Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
 - The author's moral rights;
 - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
- Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

ISBN 978-3-903337-00-8

Publisher: chemical-engineering.at

For more tutorials visit: www.cfd.at

Background

1. Why turbulence modeling?

Many engineering applications are turbulent. Turbulence is a highly transient phenomenon, characterized by a wide range of eddy sizes. One can solve these eddies numerically and obtain a full profile of the turbulent flow field. However, this is not possible as it requires a huge amount of computational effort. Hence we require a turbulence model.

An important feature in turbulence modeling is averaging, which simplifies the solution of the governing equations of turbulence. As calculating resources are limited, it is usually not possible to model the phenomena with desired grid and time resolution, so to represent scales of the flow that are not resolved by the grid models need to be applied.

There are different types of turbulence models:

- RANS-based models:
 - Linear eddy-viscosity models
 - ◆ Algebraic models
 - ◆ One and two equation models
 - Non-linear eddy viscosity models and algebraic stress models
 - Reynolds stress transport models
- Large eddy simulations
- Detached eddy simulations and other hybrid models

In this tutorial, RANS-based model is explained in detail. In the next tutorial, large eddy simulations (LES) and Smagorinsky-Lilly model will be covered.

2. RANS-based models

The governing equations for a Newtonian fluid are:

- Conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \tilde{\mathbf{u}}) = 0$$

- Conservation of momentum (Navier-Stokes equation)

$$\frac{\partial(\rho \tilde{u}_i)}{\partial t} + \nabla \cdot (\rho \tilde{u}_i \tilde{\mathbf{u}}) = -\frac{\partial \tilde{p}}{\partial x_i} + \nabla \cdot (\mu \nabla \tilde{u}_i) + \tilde{S}_{Mi}$$

- Conservation of passive scalars (given a scalar \tilde{e})

$$\frac{\partial(\rho \tilde{e})}{\partial t} + \nabla \cdot (\rho \tilde{e} \tilde{\mathbf{u}}) = \nabla \cdot (k \nabla \tilde{T}) + \tilde{S}_e$$

Note: suffix notation is used in the conservation of momentum equation for simplicity, with $i = 1$ corresponding to the x -direction, $i = 2$ the y -direction and $i = 3$ the z -direction.

One of the solutions to the problem is to reduce the number of scales (from infinity to 1 or 2) by using the Reynolds decomposition. Any property (whether a vector or a scalar) can be written as the sum of an average and a fluctuation, i.e. $\tilde{\varphi} = \Phi + \varphi$ where the capital letter denotes the average and the lower case letter denotes the fluctuation of the property. Using the Reynolds decomposition in the Navier-Stokes equations we obtain RANS or Reynolds Averaged Navier Stokes Equations.

- Average conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0$$

- Average conservation of momentum

$$\begin{aligned} \frac{\partial(\rho U_i)}{\partial t} + \nabla \cdot (\rho U_i \mathbf{U}) = & -\frac{\partial P}{\partial x_i} + \nabla \cdot (\mu U_i) - \\ & - \left(\frac{\partial(\rho \overline{u u_i})}{\partial x} + \frac{\partial(\rho \overline{v u_i})}{\partial y} + \frac{\partial(\rho \overline{w u_i})}{\partial z} \right) + S_{Mi} \end{aligned}$$

- Average conservation of passive scalars (given a scalar \tilde{e})

$$\begin{aligned} \frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\rho E \mathbf{U}) = & \nabla \cdot (k \nabla T) \\ & - \left(\frac{\partial(\rho \overline{u e})}{\partial x} + \frac{\partial(\rho \overline{v e})}{\partial y} + \frac{\partial(\rho \overline{w e})}{\partial z} \right) + S_e \end{aligned}$$

Note: a special property of the Reynolds decomposition is that the average of the fluctuating component is identically zero, a fact that is used in the derivation of the above equations.

However, by using the Reynolds decomposition, there are new unknowns that were introduced such as the turbulent stresses ($\rho \overline{u u}$, $\rho \overline{v u}$, $\rho \overline{w u}$, $\rho \overline{u v}$, $\rho \overline{v v}$, $\rho \overline{w v}$, $\rho \overline{u w}$, $\rho \overline{v w}$, $\rho \overline{w w}$) and turbulent fluxes ($\rho \overline{u e}$, $\rho \overline{v e}$, $\rho \overline{w e}$) and therefore, the RANS equations describe an open set of equations (where the over bar denotes an average). The need for additional equations to model the new unknowns is called Turbulence Modeling.

We now have 9 additional unknowns (6 Reynolds stresses and 3 turbulent fluxes). In total, for the simplest turbulent flow (including the transport of a scalar passive scalar, e.g. temperature when heat transfer is involved) there are 14 unknowns (include u , v , w , p , T)!

One possible approach to model the additional unknowns is to use the PDEs for the turbulent stresses and fluxes as a guide to modeling. The turbulent models are as follows, in order of increasing complexity:

- Algebraic (zero equation) models: mixing length (first order model)

- One equation models: k-model, μ_t -model (first order model)
- Two equation models: k- ϵ , k-kl, k- ω , low Re k- ϵ (first order model)
- Algebraic stress models: ASM (second order model)
- Reynolds stress models: RSM (second order model)
- Zero-Equation Models

In OpenFOAM®, there are two simulation types for turbulence flow, RAS and LES. As the name suggest, the RAS simulation is based on the RANS-based models covered above and will be the sole focus of this tutorial. In the next tutorial, we will move on to LES modeling and compare the results generated from these two modeling types.

simpleFoam – pitzDaily

Simulation

Use simpleFoam solver, run a steady state simulation with following turbulence models:

- kEpsilon (RAS)
- kOmega (RAS)
- LRR (RAS)

Objectives

- Understanding turbulence modeling
- Understanding steady state simulation

Data processing

Show the results of U and the turbulent viscosity in two separate contour plots.

1. Pre-processing

1.1. Copy tutorial

```
$FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily
```

1.2. 0 directory

When a turbulent model is chosen, the value of its constants and its boundary values should be set in the appropriate files. For example in kEpsilon model the k and epsilon files should be edited. See below for the epsilon file (in the 0 folder):

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

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;
    }
}
// * * * * *
* * * * *//
```

Note: Here is a list of files which should be available at 0 directory and need to be modified for each turbulence model:

- *laminar: no file*
- *kEpsilon (RAS): k and epsilon*
- *kOmega (RAS): k and omega*
- *LRR (RAS): k, epsilon and R*
- *Smagorinsky (LES): nuSgs*
- *kEqn (LES): k and nuSgs – This model is called ‘oneEqEddy’ in V2.3.0*

- *SpalartAllmaras (LES): nuSgs and nuTilda*

Some files are available, e.g. epsilon, k and nuTilda, some files should be created by the user, e.g. R, nuSgs. Templates for these files can be also found in the examples of older versions of OpenFOAM®, e.g. 1.7.1.

Note: A missing R file can be created by OpenFOAM®. Open the turbulenceProperties file in the constant directory, set the simulationType to RAS, and RASModel to kEpsilon. Run the command 'simpleFoam -postProcess -func R' from terminal, it will create the R file in the 0 directory.

OpenFOAM® v1906: it will create the turbulenceProperties:R file in the 0 directory. Then simply rename the file to R.

1.3. constant directory

In the *turbulenceProperties* file, the *simulationType* can be set as either RAS, LES or laminar. Then the corresponding sub-dictionary of the chosen simulation type needs to be defined. In this case, the sub-dictionary for RAS contains information about the chosen RAS model (kEpsilon), and the status of turbulence and *printCoeffs* are turned to on.

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

simulationType      RAS;

RAS
{
    // Tested with kEpsilon, realizableKE, kOmega, kOmegaSST, v2F,
    // ShihQuadraticKE, LienCubicKE.
    RASModel          kEpsilon;

    turbulence        on;

    printCoeffs       on;
}

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

Note: For the laminar model, set turbulence and printCoeffs to off.

1.4. system directory

Note: Since it is a steady state simulation, endTime in controlDict shows the number of iterations instead of time and deltaT should be 1, because it is the amount of increase in the iteration number.

For the LRR model, discretization model for the new variable R needs to be specified. It is done through the *fvSchemes* file,

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

ddtSchemes
```



```

{
    default            steadyState;
}

gradSchemes
{
    default            Gauss linear;
}

divSchemes
{
    default            none;
    div(phi,U)         bounded Gauss linearUpwind grad(U);
    div(phi,k)         bounded Gauss limitedLinear 1;
    div(phi,epsilon)   bounded Gauss limitedLinear 1;
    div(phi,omega)     bounded Gauss limitedLinear 1;
    div(phi,v2)        bounded Gauss limitedLinear 1;
    div(phi,R)         bounded Gauss limitedLinear 1;
    div(R)             Gauss linear;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
    div(nonlinearStress) Gauss linear;
}

laplacianSchemes
{
    default            Gauss linear corrected;
}

interpolationSchemes
{
    default            linear;
}

snGradSchemes
{
    default            corrected;
}

wallDist
{
    Method             meshWave;
}
// * * * * *
* * * * *//

```

Furthermore, *fvSolution* needs to be changed due to the new R parameter. The solver type for R is defined, in this case the solver used will be the same as the one for other variables (U, k, epsilon, omega).

2. Running simulation

```

>blockMesh
>simpleFoam

```

Note: When the solution converges, “SIMPLE solution converged in ... iterations” message will be displayed in the Shell window. If nothing happens and you do not see a message after a while (this is not the case in here, it converges after a short time), then you should check the residuals which are displayed in the Shell window manually (you should check initial residual values, it shows the difference between this iteration and the last one), if all of the Initial residual (see below) values are close to amounts you have set in the *fvSolution* then you can stop simulation (ctrl+c).

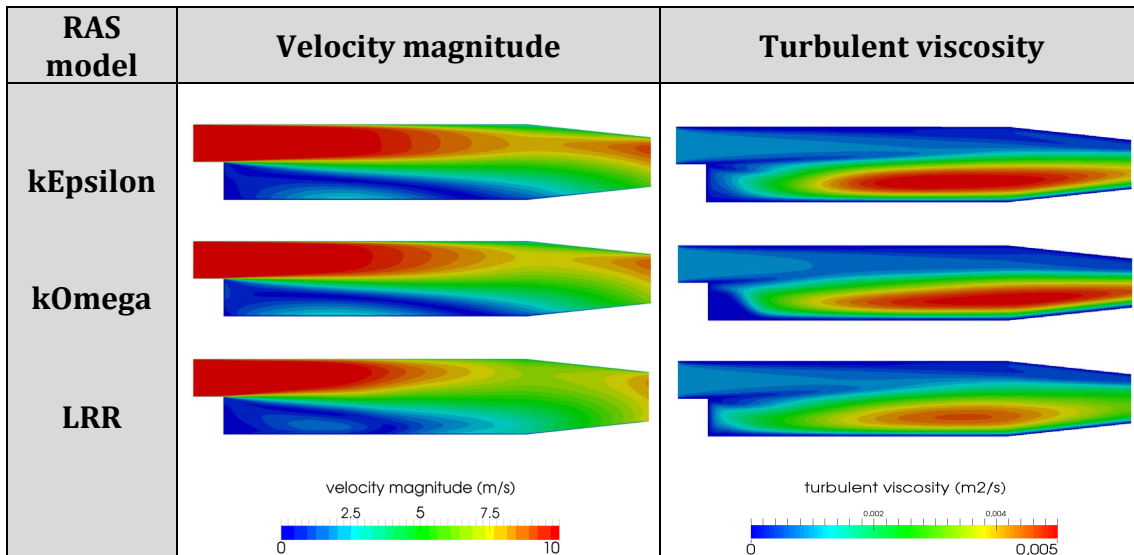
Time = 298

```
smoothSolver: Solving for Ux, Initial residual = 0.00013831, Final residual =
9.28001e-06, No Iterations 6
smoothSolver: Solving for Uy, Initial residual = 0.000977894, Final residual =
6.73868e-05, No Iterations 6
GAMG: Solving for p, Initial residual = 0.00192871, Final residual =
0.000174838, No Iterations 7
time step continuity errors : sum local = 0.000840075, global = 6.13868e-05,
cumulative = -0.193739
smoothSolver: Solving for epsilon, Initial residual = 0.000175322, Final
residual = 1.138e-05, No Iterations 2
smoothSolver: Solving for k, Initial residual = 0.000404928, Final residual =
2.99083e-05, No Iterations 2
ExecutionTime = 56.7 s ClockTime = 57 s
```

SIMPLE solution converged in 298 iterations

3. Post-processing

The simulation results are as follows (all simulations scaled to the same range):



Velocity magnitude and turbulent viscosity for different RAS models

ISBN 978-3-903337-00-8



9 783903 337008