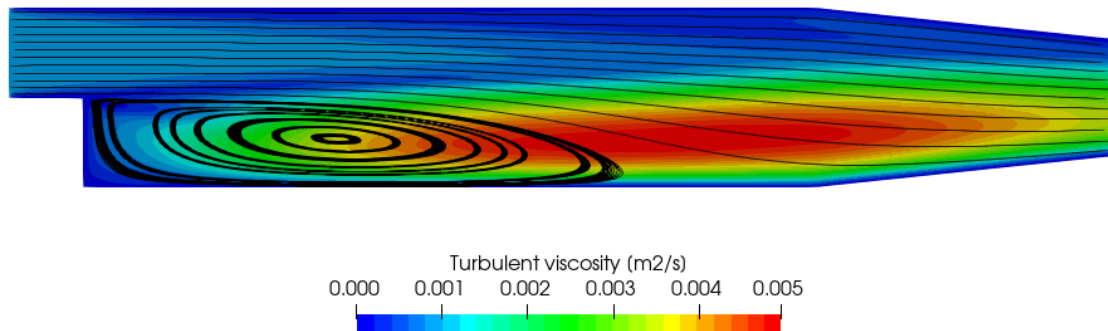


Tutorial Six


Turbulence – Steady State



Bahram Haddadi



7th edition, March 2025

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

Contributors:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek
- Clemens Gößnitzer
- Sylvia Zibuschka
- Yitong Chen
- Vikram Natarajan
- Jozsef Nagy



Technische Universität Wien
Institute of Chemical, Environmental
& Bioscience Engineering



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.

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.

Available from: www.fluidynamics.at

Background

1. Why turbulence modeling?

Many real-world engineering applications involve turbulent flow, which is a highly unsteady and chaotic phenomenon characterized by a wide range of swirling motions, called eddies, that exist at different scales. Accurately solving turbulent flows requires resolving all these eddies, which would demand an enormous amount of computational power and memory. In practical applications, such a Direct Numerical Simulation (DNS) is infeasible due to its excessive computational cost.

To overcome this challenge, turbulence models are used to approximate the effects of turbulent eddies without resolving them explicitly. These models simplify the governing equations of fluid flow while still capturing the essential characteristics of turbulence.

An important principle in turbulence modeling is averaging, which simplifies the governing equations of turbulent motion. Due to computational limitations, it is not always possible to model turbulent flow at fine spatial and temporal resolutions. Turbulence models compensate for this limitation by representing the unresolved scales of motion.

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})

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

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

$$\frac{\partial(\rho U_i)}{\partial t} + \nabla \cdot (\rho U_i \mathbf{U}) = -\frac{\partial P}{\partial x_i} + \nabla \cdot (\mu U_i) - \left(\frac{\partial(\rho \overline{uu_i})}{\partial x} + \frac{\partial(\rho \overline{vu_i})}{\partial y} + \frac{\partial(\rho \overline{wu_i})}{\partial z} \right) + S_{Mi}$$

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

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

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 ($\rho \overline{uu}$, $\rho \overline{vu}$, $\rho \overline{wu}$, $\rho \overline{uv}$, $\rho \overline{vv}$, $\rho \overline{vw}$, $\rho \overline{uw}$, $\rho \overline{vw}$, $\rho \overline{ww}$) and turbulent fluxes ($\rho \overline{u\tilde{e}}$, $\rho \overline{v\tilde{e}}$, $\rho \overline{w\tilde{e}}$) 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.

incompressibleFluid – pitzDaily

Tutorial outline

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

- kEpsilon (RAS)
- kOmega (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. Copying tutorial

Copy the following tutorial to your working directory:

```
$FOAM_TUTORIALS/incompressibleFluid/pitzDaily
```

Replace the system directory with the system directory from the following tutorial:

```
$FOAM_TUTORIALS/incompressibleFluid/pitzDailySteadyExperimentalInlet
```

Copy the *pitzDaily* file for the pitzDaily geometry from following directory to your system directory:

```
$FOAM_TUTORIALS/resources/blockMesh
```

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): s
- kEqn (LES): k and s
- 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.

1.3. constant directory

In the *momentumTransport* 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. Setting the `turbulence` to `off/false` will turn the turbulence model off and perform a laminar simulation.

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

simulationType      RAS;

RAS
{
    model            kEpsilon;

    turbulence       on;

    printCoeffs      on;
}

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

Note: For Boolean inputs in OpenFOAM either on/off, 1/0 or true/false can be used.

1.4. system directory

Running simulations in steady state mode, the `endTime` in the *controlDict* file corresponds to maximum number of iterations (e.g. 1000) instead of time, `deltaT` is the iterator and should be 1, because it is the amount of increase in the iteration number and `writeInterval` will show the frequency of saving data (e.g. 50 means each 50 iterations a saving will be done).

In the *fvSchemes* files the `ddtSchemes` is set to `steadyState` which will set the time derivative part of conservation equations to zero which is compatible with the steady state assumption.

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

ddtSchemes
{
    default          steadyState;
}

gradSchemes
{
    default          Gauss linear;
}

divSchemes
{
    default          none;
    div(phi,U)       bounded Gauss Upwind;
    div(phi,k)       bounded Gauss Upwind;
    div(phi,epsilon) bounded Gauss Upwind;
    div(phi,R)       bounded Gauss Upwind;
    div(R)           Gauss linear;
    div(phi,nuTilda) bounded Gauss Upwind;

    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}

laplacianSchemes
{
    default          Gauss linear corrected;
}

interpolationSchemes
{
    default          linear;
}

snGradSchemes
{
    default          corrected;
}
// * * * * *
* * * * *//
```

In case we are solving for property such as *omega* and it is not defined in the schemes and there is no default schemes defined, it should be added to the relevant schemes section, e.g. to the `divSchemes` (`div(phi,omega)`). Also, *fvSolution* needs to be adopted based on the new parameters, e.g. *omega*.

Note: In the *fvSolution* the solver type and settings need to be defined or be added to the others, e.g. for *omega* "(U|k|epsilon|R|nuTilda|omega)".

2. Running simulation

```
>blockMesh -dict system/pitzDaily
```

Note: The default dictionary file for *blockMesh* is *blockMeshDict* in the *system* directory. It is possible to use a different dictionary file by using the flag "-dict" and the address of the file (dictionary), in this case *system/pitzDaily*.

```
>foamRun -solver incompressibleFluid
```

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).

Note: You can use bash script and gnuPlot for extracting the residual data from log files and plotting them. For saving the simulation output to a log file use the following command for running the simulation and the terminal output will be saved to the log file (log file can be viewed using less - check Appendix A).

```
>foamRun -solver incompressibleFluid > log
```

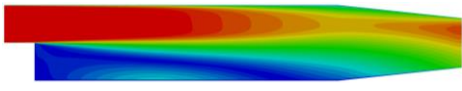
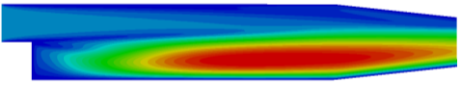
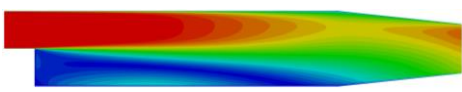
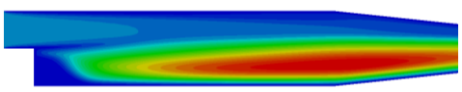
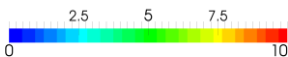
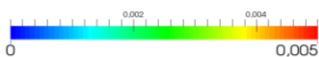
```
Time = 795s
```

```
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 795 iterations
```

3. Post-processing

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

RAS model	Velocity magnitude	Turbulent viscosity
kEpsilon		
kOmega		
	velocity magnitude (m/s) 	turbulent viscosity (m2/s) 

Velocity magnitude and turbulent viscosity for different RAS models