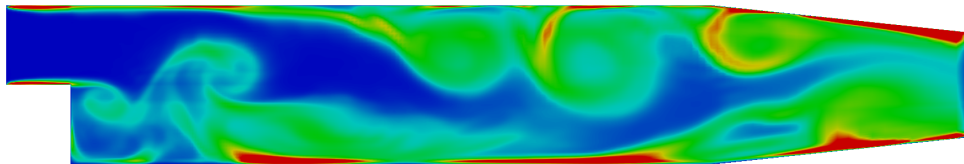


# Tutorial Seven

## Turbulence - Transient



5<sup>th</sup> edition, Sep. 2019



ICEBE  
IMAGINEERING  
NATURE



WARDS  
openFOAM<sup>®</sup>  
cfd.at

This offering is not approved or endorsed by ESI<sup>®</sup> Group, ESI-OpenCFD<sup>®</sup> or the OpenFOAM<sup>®</sup> Foundation, the producer of the OpenFOAM<sup>®</sup> software and owner of the OpenFOAM<sup>®</sup> 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](http://www.cfd.at)**

## Background

### 1. Large eddy simulation (LES)

In LES models, it is assumed that large eddies of the flow are dependent on the geometry, while the smaller eddies are more universal. One can then explicitly solve for the large eddies in a calculation by resolving them in a grid and implicitly account for the small eddies by using a sub-grid-scale model (SGS model).

Mathematically, it is like separating the velocity field into a resolved and sub-grid part using a filter function. The resolved part of the field represents the large eddies, while the sub-grid part of the velocity represents the small eddies whose effect on the resolved field is included through the sub-grid-scale model. Formally, one may think of filtering as the convolution of a function with a filtering kernel  $G$ :

$$\bar{u}_i(\vec{x}) = \int G(\vec{x} - \vec{\xi}) u(\vec{\xi}) d\vec{\xi}$$

resulting in

$$u_i = \bar{u}_i + w_i$$

Where  $\bar{u}_i$  is the resolvable scale part and  $w_i$  is the subgrid-scale part. However, most practical (and commercial) implementations of LES use the grid itself as the filter and perform no explicit filtering. The filtered equations are developed from the incompressible Navier-Stokes equations of motion:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left( \nu \frac{\partial u_i}{\partial x_j} \right)$$

Substituting in the decomposition  $u_i = \bar{u}_i + w_i$  and  $p = \bar{p} + p'$  and then filtering the resulting equation gives the equations of motion for the resolved field:

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j} \left( \nu \frac{\partial \bar{u}_i}{\partial x_j} \right) + \frac{1}{\rho} \frac{\partial \tau_{ij}}{\partial x_j}$$

We have assumed that the filtering operation and the differentiation operation commute, which is not generally the case. It is thought that the errors associated with this assumption are usually small, though filters that commute with differentiation have been developed. The extra term  $\partial \tau_{ij} / \partial x_j$  arises from the non-linear advection terms, due to the fact that:

$$u_j \frac{\partial u_i}{\partial x_j} \neq \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j}$$

and hence

$$\tau_{ij} = \bar{u}_i \bar{u}_j - \overline{u_i u_j}$$

Similar equations can be derived for the sub grid-scale field. Sub grid-scale turbulence models usually employ the Boussinesq hypothesis, and seek to calculate (the deviatoric part of) the SGS stress using:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\mu_t\bar{S}_{ij}$$

where  $\bar{S}_{ij}$  is the rate-of-strain tensor for the resolved scale defined by

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right)$$

and  $\nu_t$  is the subgrid-scale turbulent viscosity. Substituting into the filtered Navier-Stokes equations, we then have:

$$\frac{\partial\bar{u}_i}{\partial t} + \bar{u}_j\frac{\partial\bar{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left([\nu + \nu_t]\frac{\partial\bar{u}_i}{\partial x_j}\right)$$

where we have used the incompressibility constraint to simplify the equation and the pressure is now modified to include the trace term  $\tau_{kk}\delta_{ij}/3$ .

## 2. Smagorinsky-Lilly model

A simple model for Sub grid-scale model is the Smagorinsky model, which can be summarized as:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2(C_s\Delta)^2|\bar{S}|S_{ij}$$

In the Smagorinsky-Lilly model, the eddy viscosity is modeled by

$$\mu_{sgs} = \rho(C_s\Delta)^2|\bar{S}|$$

Where the filter width is usually taken to be

$$\Delta = (Volume)^{1/3}$$

and

$$|\bar{S}| = \sqrt{2S_{ij}S_{ij}}$$

The effective viscosity is calculated from

$$\mu_{eff} = \mu_{mol} + \mu_{sgs}$$

The Smagorinsky constant usually has the value:  $C_s = 0.1 - 0.2$

## **pisoFoam – pitzDaily**

### **Simulation**

Use the pisoFoam solver, run a backward facing step case for 0.2 s with different turbulence models:

- Smagorinsky (LES)
- kEqn (LES)
- kEpsilon (RAS)

### **Objectives**

- Understanding turbulence models
- Understanding the difference between transient and steady state simulation
- Finding appropriate turbulence model

### **Data processing**

Display the results of U and the turbulent viscosity in two separate contour plots at three different time steps. Compare with steady state simulation (Tutorial Six).

## 1. Pre-processing

### 1.1. Copy tutorial

Copy the tutorial from the following directory to your working directory:

```
$FOAM_TUTORIALS/incompressible/pisoFoam/LES/pitzDaily
```

### 1.2. 0 directory

Set the turbulence model initial and boundary values.

*Note: For different turbulent models, different files should be modified (check Tutorial Six).*

### 1.3. constant directory

As mentioned in Tutorial Six, in *turbulenceProperties* the turbulent model type has to be set. The *simulationType* can be changed to `LES` or `RAS`. Depending on which type is selected, the corresponding sub-dictionary needs to be specified. Below is the *turbulenceProperties* file for the `kEqn` model which is an LES model.

```
// * * * * *
* * * * *//
simulationType LES;

LES
{
    LESModel          kEqn;
    turbulence        on;
    printCoeffs      on;
    delta             cubeRootVol;

    dynamicKEqnCoeffs
    {
        filter        simple;
    }

    cubeRootVolCoeffs
    {
        deltaCoeff    1;
    }

    PrandtlCoeffs
    {
        delta          cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff    1;
        }

        smoothCoeffs
        {
            delta          cubeRootVol;
            cubeRootVolCoeffs
            {
                deltaCoeff    1;
            }

            maxDeltaRatio  1.1;
        }

        Cdelta          0.158;
    }

    vanDriestCoeffs
```

```

    {
        delta            cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff    1;
        }

        smoothCoeffs
        {
            delta            cubeRootVol;
            cubeRootVolCoeffs
            {
                deltaCoeff    1;
            }

            maxDeltaRatio    1.1;
        }

        Aplus            26;
        Cdelta            0.158;
    }

    smoothCoeffs
    {
        delta            cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff    1;
        }

        maxDeltaRatio    1.1;
    }
}
// * * * * *
* * * * *//

```

## 2. Running simulation

```

>blockMesh
>pisoFoam

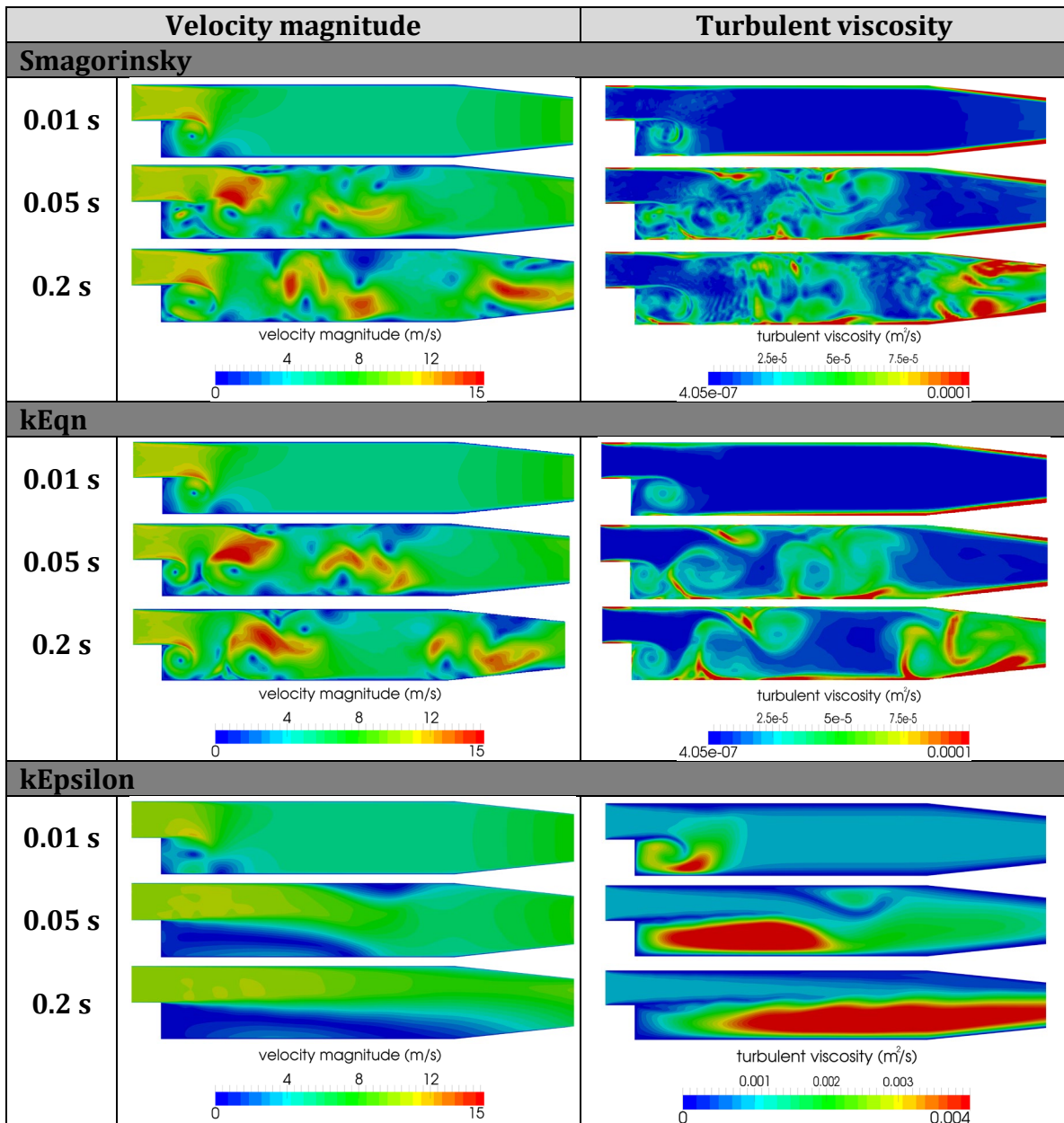
```

## 3. Post-processing

The simulation results are as follows:

For the kEpsilon model after 0.2s the results are similar to the steady state simulation. Therefore, it can be assumed it has reached the steady state. Other models do not have a steady situation and are fluctuating all the time, so they require averaging for obtaining steady state results.

kEpsilon and other RAS models use averaging to obtain the turbulence values, but LES does not include any averaging by default. Therefore, LES simulations should use a higher grid resolution (smaller cells) and smaller time steps (for reasonable Co number). Contour plots or other LES results should be presented time averaged over reasonable number of time steps (not done in this tutorial).



Comparison of different turbulent models for transient simulation.



ISBN 978-3-903337-00-8



9 783903 337008