# Mandatory Assignment 1

Jørgen D. Tyvand

October 1, 2015

# 1)

(The following description of pisoFoam is largely based on the description given on openfoamwiki.net)

The basic structure of the piso algorithm as implemented in pisoFoam is as follows:

**Step1**: Boundary conditions are set

**Step2**: The discretized momentum equation is defined and solved:

```
fvVectorMatrix UEqn
            (
                fvm::ddt(U)
              + fvm::div(phi, U)
              + turbulence->divDevReff(U)
            );
```

This equation differs in my version of OpenFoam from the one given for pisoFoam at openfoamwiki.net and Chalmers in the final term, where the other sources gives

```
 fvVectorMatrix UEqn
 (
        fvm::ddt(U)
      + fvm::div(phi, U)
      - fvm::laplacian(nu,Step 2 U)
 );
```

The final term in my version, as i understand, is a combination of a laplacian and the divergence of the deviatoric (the deviatoric being the difference between the Cauchy stress tensor and the hydrostatic stress tensor)

```
divDevReff(U) =
- fvm::laplacian(nuEff(), U)
- fvc::div(nuEff()*dev(fvc::grad(U)().T()))
```

We then solve the equation

```
solve(UEqn == -fvc::grad(p));
```

**Step3**: The piso loop then starts, begining with calculating the coefficients and the flux

```
 volScalarField rAU(1.0/UEqn.A());

volVectorField HbyA("HbyA", U);
HbyA = rAU*UEqn.H();
surfaceScalarField phiHbyA
 (
          "phiHbyA",
           (fvc::interpolate(HbyA) & mesh.Sf())
            + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
```

```
);

adjustPhi(phiHbyA, U, p);
```

**Step4**: The pressure equation is solved

```
fvScalarMatrix pEqn
 (
     fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
 );

pEqn.setReference(pRefCell, pRefValue);
```

In this case, the nNonOrthCorr is set to 0, so there are no non-orthogonal pressure corrections.

**Step5**: The velocites and boundary conditions are corrected

```
 U = HbyA - rAU*fvc::grad(p);
U.correctBoundaryConditions();
```

**Step6**: The loop is repeated the number of times given by nCorrectors in the fvSolution file.

.

**Step7**: The time step is increased and the process starts again from step 1.

## 2)

For both the LES and RANS models, we start from the Navier-Stokes equations for incompressible flow:

$\nabla \cdot (\rho \mathbf{u}) = 0$

$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \mathbf{u}) = -\frac{\partial p}{\partial x} + \mu \nabla^2 u$

$\frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \mathbf{u}) = -\frac{\partial p}{\partial y} + \mu \nabla^2 v$

Lengthy derivations of the LES and RANS equations will not be given, but a short explanation of each will give the general process.

For Large Eddy Simulation (LES), we use spatial filtering to separate varying sizes of eddies. A cutoff width $\Delta$ is introduced, for which information about eddies smaller than the given width will be ignored/destroyed. A spatial filtering using a filter finction $G(\mathbf{x}, \mathbf{x}', \Delta)$ is introduced, giving in the following form (3.84 in the book):

$$\overline{\phi}(\mathbf{x}, t) \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(\mathbf{x}, \mathbf{x}', \Delta) \phi(\mathbf{x}', t) \mathrm{d}x_1' \mathrm{d}x_2' \mathrm{d}x_3'$$

where $\overline{\phi}(\mathbf{x}, t)$ and $\phi(\mathbf{x}, t)$ are the filtered and unfiltered functions respectively. The filter function $G(\mathbf{x}, \mathbf{x}', \Delta)$ can be given in several ways, but the one used in finite volume

implementations is the top-hat/box filter function

$$G(\mathbf{x}, \mathbf{x}', \Delta) = \begin{cases} \frac{1}{\Delta^3} & |\mathbf{x} - \mathbf{x}'| \leq \Delta/2 \\ 0 & |\mathbf{x} - \mathbf{x}'| > \Delta/2 \end{cases}$$

Using this filtering on the Navier-Stokes equations, we get the LES momentum equations (the intermediate step from 3.88a-c to 3.89a-c in the book for rewriting the term $\nabla \cdot (\rho\overline{\phi\mathbf{u}})$ is not shown):

$$\frac{\partial(\rho\overline{u})}{\partial t} + \nabla \cdot (\rho\overline{u}\,\overline{\mathbf{u}}) = -\frac{\partial \overline{p}}{\partial x} + \mu\nabla^2\overline{u} - (\nabla \cdot (\rho\overline{u\mathbf{u}}) - \nabla \cdot (\rho\overline{u}\,\overline{\mathbf{u}}))$$

$$\frac{\partial(\rho\overline{v})}{\partial t} + \nabla \cdot (\rho\overline{v}\,\overline{\mathbf{u}}) = -\frac{\partial \overline{p}}{\partial y} + \mu\nabla^2\overline{v} - (\nabla \cdot (\rho\overline{v\mathbf{u}}) - \nabla \cdot (\rho\overline{v}\,\overline{\mathbf{u}}))$$

The boundary conditions for the LES problem is as following:

Uniform velocity of 10 m/s in the x-direction at the inlet
Zero velocity gradient at the walls and outlet
Zero pressure gradient at the walls and inlet
A uniform value of 0 for the pressure at the outlet

For the simulations i have used 2 different mesh refinements, one with 10x20 cells for the inlet and outlet boxes and 100x20 for the center boxes, as well as a doubled mesh of 20x40 and 200x40 boxes. The case files were originally copied from the PitzDaily case for incompressible flow with PISO, and edited from there. For the convection term i have tested different upwind schemes, and landed on filteredLinear. An upwind scheme will include more information from upwind cells, and therefore (hopefully) give a more correct and stable calculation. I have tested the following 4 schemes for the coarsest mesh:

Non-upwind linear (as found in the unaltered PitzDaily files)
upwind
linearUpwind
filteredLinear

The following four movie files shows a simulation of these four schemes for 0.5 seconds on the 100 mesh:

(a) **linear** (b) **upwind** (c) **upwindLinear** (d) **filteredLinear**

The filteredLinear scheme has also been used for the 200 mesh, as well as a even finer mesh of 250x50 central boxes. As we can see from the following videos, the solutions are

mot mesh independent.

I also found that the solver crashed if i used a too fine mesh with a too coarse time step. In the final calculations i have used timesteps of $1.0 * 10^{-5}$, but with a timestep of $1.0 * 10^{-4}$ the Courant number grew to between 1.5 and 2, at which the calculations crashed. This implies grid sensitivity with respect to grid size and time step size.

## 3)

For the RANS models i have chosen to use the simpleFoam solver for the $k-\epsilon$ and $k-\omega$ models. The general RANS equations are as follows:

Here the overline marks the mean terms, and the marked terms are the fluctuation terms.

For the $k-\epsilon$ model, I have used the value for k that was used in the PitzDaily case. This is because the inlet velocity is the same, and i have assumed that a similar turbulence intensity is appropriate. Analyzing the value for $\epsilon$ in the PitzDaily case, i found that they have used a value of 0.1 times the inlet opening for the turbulent length scale. I have adjusted my value for $\epsilon$ using the same ratio.

There are two additional equations to be solved for the $k-\epsilon$ model:

$$\frac{\partial \rho k}{\partial t} + \nabla \cdot (\rho k \mathbf{u}) = \frac{\mu_t}{\sigma_k}\nabla^2 k + 2\mu_t S_{ij} \cdot S_{ij} - \rho\epsilon$$

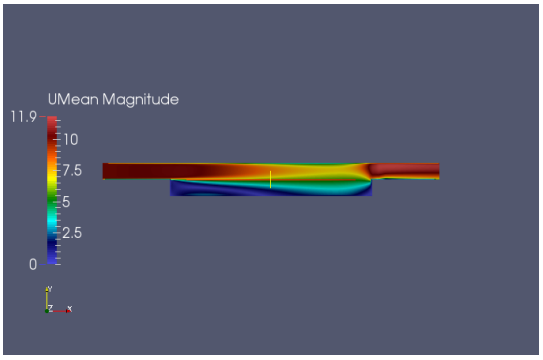
$$\frac{\partial \rho \epsilon}{\partial t} + \nabla \cdot (\rho \epsilon \mathbf{u}) = \frac{\mu_t}{\sigma_\epsilon}\nabla^2 \epsilon + C_{1\epsilon}\frac{\epsilon}{k}2\mu_t S_{ij} \cdot S_{ij} - C_{2\epsilon}\rho\frac{\epsilon^2}{k}$$

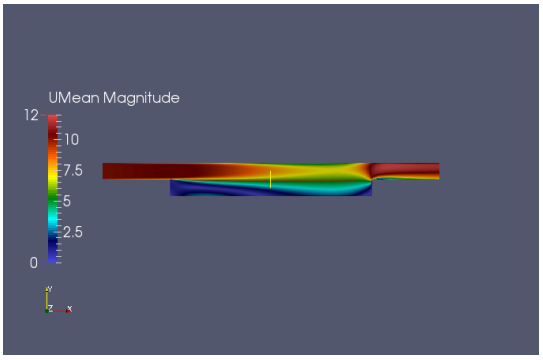
Here, $S_{ij}$ are the components of the mean of the rate of deformation. The constants have the following given values (found both in the book and in other sources):

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} \qquad C_\mu = 0.09 \qquad \sigma_k = 1.00 \qquad \sigma_\epsilon = 1.30 \qquad C_{1\epsilon} = 1.44 \qquad C_{2\epsilon} = 1.92$$

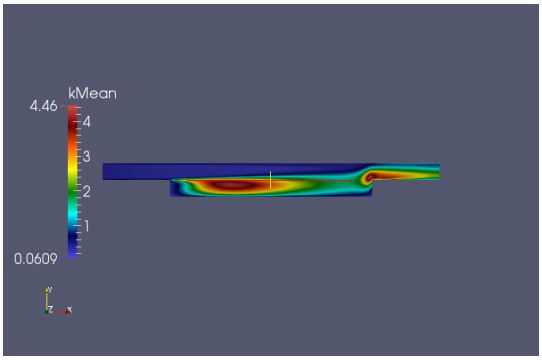
Running the $k-\epsilon$ case, i have found the following converged solutions for the flow, using the 100 and 200 meshes respectively:

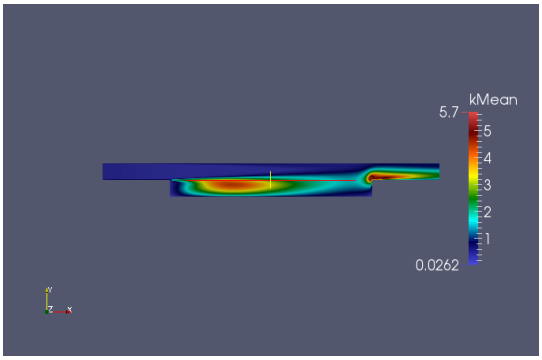
The following figures show the mean velocity for the two mesh sizes:




And finally we have the mean kinematic energy:
We see that there is virtually no difference between the two cases, so there is little or no




mesh sensitivity using simpleFoam for this RANS problem.

For the $k-\omega$ model, i have used the fact that $\omega = \frac{\epsilon}{k}$, and used the values from the $k-\epsilon$ problem to calculate $\omega$. We have two additional equations in this model as well:

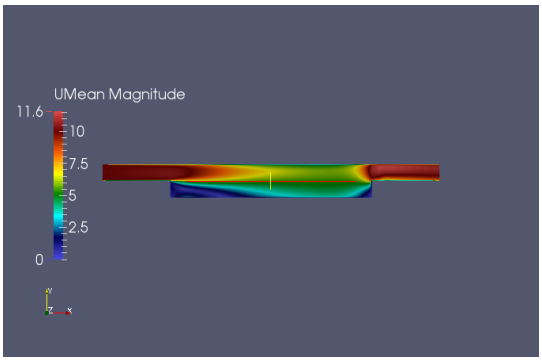
$$\frac{\partial \rho k}{\partial t} + \nabla \cdot (\rho k \mathbf{u}) = (\mu + \frac{\mu_t}{\sigma_k})\nabla^2 k + P_k - \beta^* \rho k \omega \qquad P_k = \left(2\mu_t S_{ij} \cdot S_{ij} - \frac{2}{3}\rho k \frac{\partial U_i}{\partial x_j}\delta_{ij}\right)$$

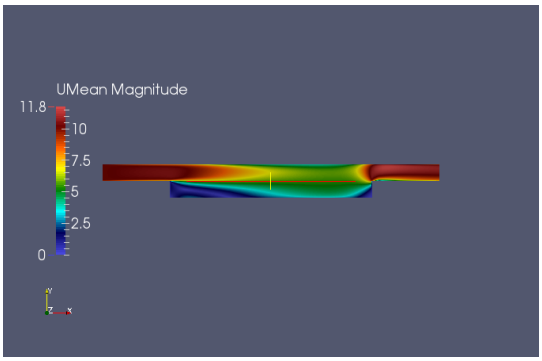
$$\frac{\partial \rho \omega}{\partial t} + \nabla \cdot (\rho \omega \mathbf{u}) = (\mu + \frac{\mu_t}{\sigma_\omega})\nabla^2 \omega + \gamma_i \left(2\rho S_{ij} \cdot S_{ij} - \frac{2}{3}\rho\omega \frac{\partial U_i}{\partial x_j}\delta_{ij}\right) - \beta_1 \rho \omega^2$$

With constants

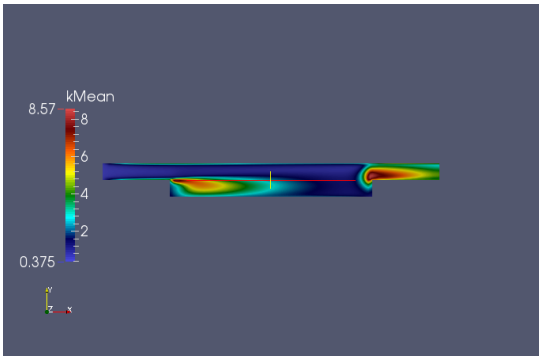
$$\sigma_k = 2.0 \qquad \sigma_\omega = 2.0 \qquad \gamma_1 = 0.553 \qquad \beta_1 = 0.075 \qquad \beta^* = 0.09$$

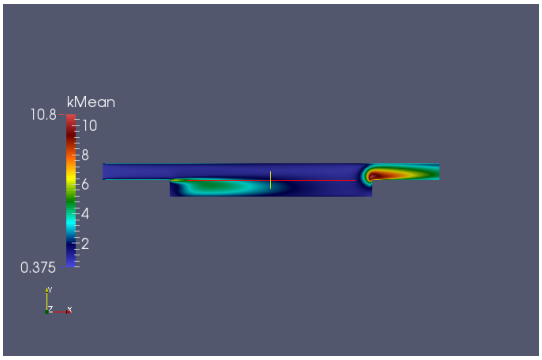
Running simpleFoam, we get the following plots for the converged values of the mean velocity for the 100 and 200 meshes:




And for the mean kinematic energy:




## 4)

## 5)

## 6)

Since turbulence is a three-dimensional phenomenon, I would assume that some critical information could be lost using LES as a 2D model. One example could be an eddy with primarily extension in the z-direction, that might have a width below the cutoff value in the x- or y-directions. Thus the impact of this eddy on the mean flow, which might be significant, could potentially be lost by using only a 2D approach.