How to implement a turbulence model
How to implement your own turbulence model (1/3)

- The implementations of the turbulence models are located in
  $FOAM_SRC/turbulenceModels

- Copy the source of the turbulence model that is most similar to what you want to do. In this case we will make our own copy of the kOmegaSST turbulence model and create a directory structure as in the OpenFOAM installation:

  ```
  cd $WM_PROJECT_DIR
  cp -r --parents src/turbulenceModels/incompressible/RAS/kOmegaSST \ $WM_PROJECT_USER_DIR
  cd $WM_PROJECT_USER_DIR/src/turbulenceModels/incompressible/RAS
  mv kOmegaSST mykOmegaSST
  ```
How to implement your own turbulence model (2/3)

- We also need a Make/files and a Make/options (c.f. $FOAM_SRC/turbulenceModels/incompressible/RAS/Make)

- Create a Make directory:
  
  mkdir Make

- Create Make/files (we are only adding mykOmegaSST):
  
  echo "mykOmegaSST/mykOmegaSST.C"
  LIB = "$FOAM_USER_LIBBIN)/libmyIncompressibleRASModels" > Make/files

- Create Make/options:
  
  echo "EXE_INC = \"
  -I"$(LIB_SRC)/turbulenceModels \"
  -I"$(LIB_SRC)/transportModels \"
  -I"$(LIB_SRC)/finiteVolume/lnInclude \"
  -I"$(LIB_SRC)/meshTools/lnInclude \"
  -I"$(LIB_SRC)/turbulenceModels/incompressible/RAS/lnInclude
  LIB_LIBS =" > Make/options

  (the last -I is needed since mykOmegaSST uses include-files in the original directory)
How to implement your own turbulence model (3/3)

- We need to modify the file names of our new turbulence model:

  cd mykOmegaSST; rm kOmegaSST.depmv kOmegaSST.C mykOmegaSST.C; mv kOmegaSST.H mykOmegaSST.H

- In mykOmegaSST.C and mykOmegaSST.H, change all occurrences of kOmegaSST to mykOmegaSST so that we have a new class name:

  sed -i s/kOmegaSST/mykOmegaSST/g mykOmegaSST.C
  sed -i s/kOmegaSST/mykOmegaSST/g mykOmegaSST.H

- Introduce a small modification so that we can see if we use our new model. Add within the curly brackets of the constructor in mykOmegaSST.C:

  Info << "Defining my own kOmegaSST model" << endl;

- Compile using:

  cd ..; wmake libso

  which will build a dynamic library.
Test on the simpleFoam/pitzDaily tutorial

We will use our turbulence model on the simpleFoam/pitzDaily tutorial:

```
run
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily .
cd pitzDaily
blockMesh

continued...
```
How to use your own turbulence model

• Tell OpenFOAM to use your new library by adding a line to \texttt{controlDict}:
  \begin{verbatim}
  libs ("libmyIncompressibleRASModels.so");
  \end{verbatim}

• You choose turbulence model in the \texttt{constant/RASProperties} dictionary:
  \begin{verbatim}
  RASModel mykOmegaSST;
  \end{verbatim}

• You also need to generate a \texttt{0/omega} file, and update \texttt{fvSchemes} and \texttt{fvSolution}
  \begin{verbatim}
  cp 0/epsilon 0/omega
sed -i s/epsilon/omega/g 0/omega
sed -i s/"0 2 -3 0 0 0 0"/"0 0 -1 0 0 0 0"/g 0/omega
  \end{verbatim}
  \begin{verbatim}
  sed -i s/14.855/440.15/g 0/omega
  sed -i s/epsilon/omega/g system/fvSchemes
  sed -i s/epsilon/omega/g system/fvSolution
  \end{verbatim}

  (here $\nu_t = \frac{k}{\omega} = C_\mu \frac{k^2}{\varepsilon}$ is kept the same in both cases)

• Now you can run the \texttt{simpleFoam/pitzDaily} tutorial with your new turbulence model. Try both \texttt{kOmegaSST} and \texttt{mykOmegaSST} and look for your write-statement in the log file.

• Simply add appropriate source terms to implement a variant of \texttt{kOmegaSST}....
A note on new libraries

• It is a common habit to make backup copies of directories when doing new implementations. This may cause problems when implementing libraries.

• One of the steps when compiling a library with `wmake` is to create the `lnInclude` directory. In that process all of the sub-directories, to the directory where the `Make` directory is located, are searched for files. If you have a backup copy of a directory, you have two files with the same name, and you thus do not know which one will be linked to in `lnInclude`.

• You can still do backups, but then pack up the directory with `tar czf <directory>.tgz <directory>` and remove the original directory.
A note on new turbulence models

- The RAS turbulence models in OpenFOAM are sub-classes to the virtual class RASModel.

- You are only allowed to use the same member function definitions as in the RASModel class. If you need other member functions you will have to add those to the RASModel class, which requires that you copy and modify all of $FOAM_SRC/turbulenceModels/incompressible/RAS$. You can recognize where the top-level of a class is located by locating the Make-directory.

We will now have a look at the implementation of the kOmegaSST model.
$k - \omega$ SST in OpenFOAM-1.6 (almost identical in newer)

From $FOAM_SRC/turbulenceModels/incompressible/RAS/kOmegaSST/kOmegaSST.H$:

- Menter, F., Esch, T.
  "Elements of Industrial Heat Transfer Prediction"
  16th Brazilian Congress of Mechanical Engineering (COBEM), Nov. 2001

- Note that this implementation is written in terms of alpha diffusion coefficients rather than the more traditional sigma (alpha = 1/sigma) so that the blending can be applied to all coefficients in a consistent manner. The paper suggests that sigma is blended but this would not be consistent with the blending of the k-epsilon and k-omega models.

- Also note that the error in the last term of equation (2) relating to sigma has been corrected.

- Wall-functions are applied in this implementation by using equations (14) to specify the near-wall omega as appropriate.

- The blending functions (15) and (16) are not currently used because of the uncertainty in their origin, range of applicability and that is $y+$ becomes sufficiently small blending $u_{\tau}$ in this manner clearly becomes nonsense.
$k - \omega$ SST: Equations

\[
\begin{align*}
\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} &= P_k - \beta^* k \omega + \frac{\partial}{\partial x_j} \left[ (\nu + \sigma_k \nu_l) \frac{\partial k}{\partial x_j} \right] \\
\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} &= \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[ (\nu + \sigma_\omega \nu_l) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F_1)\sigma_\omega^2 \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \\
\nu_l &= \frac{a_1 k}{\max(a_1 \omega, SF_2)}, \quad P_k = \min(G, 10 \beta^* k \omega), \quad G = \nu_t \frac{\partial U_i}{\partial x_j} \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \\
S^2 &= \left| \frac{1}{2} (\partial_j u_i + \partial_i u_j) \right|^2, \quad S = \sqrt{S^2} = \left| \frac{1}{2} (\partial_j u_i + \partial_i u_j) \right| \\
F_1 &= \tanh \left\{ \min \left( \min \left[ \max \left( \frac{\sqrt{k}}{\beta^* \omega y}, \frac{500 \nu}{y^2 \omega} \right), \frac{4 \sigma_\omega^2 k}{CD_{k\omega} y^2} \right], 10 \right) \right\}^4 \\
F_2 &= \tanh \left[ \min \left( \max \left( \frac{2 \sqrt{k}}{\beta^* \omega y}, \frac{500 \nu}{y^2 \omega} \right), 100 \right) \right]^2 \\
CD_{k\omega} &= 2\sigma_\omega^2 \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i}
\end{align*}
\]

Parameters that are blended using $F_1$:

$\sigma_k, \sigma_\omega, \alpha, \beta$

Constants:

$\beta^*, \sigma_{k1}, \sigma_{k2}, \sigma_{\omega1}, \sigma_{\omega2}, \alpha_1, \alpha_2, \beta_1, \beta_2, a_1$
$k - \omega$ SST in OpenFOAM-1.6, $\nu_t$

**Source code:**

```
$FOAM_SRC/turbulenceModels/incompressible/RAS/kOmegaSST
```

**Kinematic eddy viscosity:**

$$\nu_t = \frac{a_1 k}{\max(a_1 \omega, SF_2)}$$

**kOmegaSST.C:**

```cpp
nut_ =
    a1_*k_/max(a1_*(omega_ + omegaSmall_), F2() *mag(symm(fvc::grad(U_))));
```

In **kOmegaSST.C**:

```cpp
    a1_(dimensioned<scalar>::lookupOrAddToDict("a1",coeffDict_,0.31))
```

In **kOmegaSST.C (S = sqrt(S2)):**

```cpp
volScalarField S2 = magSqr(symm(fvc::grad(U_)));
```

i.e. $S^2 = \left| \frac{1}{2} \left( \partial_j u_i + \partial_i u_j \right) \right|^2$ and $S = \sqrt{S^2} = \left| \frac{1}{2} \left( \partial_j u_i + \partial_i u_j \right) \right|$.

F2() is a blending function, which is described on the next slide.
\( k - \omega \) SST in OpenFOAM-1.6, F2()

F2() is a blending function:

\[
F_2 = \tanh \left[ \min \left( \max \left( \frac{2 \sqrt{k}}{\beta \omega y}, \frac{500 \nu}{y^2 \omega} \right), 100 \right) \right]^2
\]

In `kOmegaSST.C`:

```cpp
tmp<volScalarField> kOmegaSST::F2() const
{
    volScalarField arg2 = min
    (,
     max
     (,
      (scalar(2)/betaStar_)*sqrt(k_)/(omega_*y_),
      scalar(500)*nu_/(sqr(y_)*omega_))
    ),
    scalar(100)
    );

    return tanh(sqr(arg2));
}
```
\( k - \omega \) SST in OpenFOAM-1.6, Turbulence kinetic energy eq.

\[
\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P_k - \beta^*k\omega + \left( \nu + \sigma_k\nu_t \right) \frac{\partial k}{\partial x_j} + \frac{\partial}{\partial x_j} \left[ \left( \nu + \sigma_k\nu_t \right) \frac{\partial k}{\partial x_j} \right], \quad P_k = \min(G, 10\beta^*k\omega), \quad G = \nu_t \frac{\partial U_i}{\partial x_j} \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right)
\]

In \texttt{kOmegaSST.C}:

\[
\text{tmp<fvScalarMatrix> kEqn} (\
\quad \text{fvm::ddt (k_)} \\
\quad + \text{fvm::div (phi_, k_)} \\
\quad - \text{fvm::Sp (fvc::div (phi_), k_)} \\
\quad - \text{fvm::laplacian (DkEff (F1), k_)} \\
\quad =\min (G, c1_\ast betaStar\_\ast k\_\ast omega\_) \\
\quad - \text{fvm::Sp (betaStar\_\ast omega\_, k\_)} \\
\)
\]

The effective diffusivity for \( k \), \((DkEff (F1))\), is described on a later slide.

F1 is obtained from \texttt{F1()}\), which is a blending function for \( \sigma_k \), and is described on the next slide,

where \( CD_{kw} = 2 \sigma_\omega^2 \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \)

\[
\text{volScalarField CDkOmega =} \\
(2*alphaOmega2\_)*(fvc::grad(k_) & fvc::grad(omega_))/omega_; \\
\]

©Håkan Nilsson, Chalmers / Applied Mechanics / Fluid Dynamics 13
F1() is a blending function, kOmegaSST.C (compressed here):

\[
F_1 = \tanh \left\{ \left\{ \min \left( \min \left[ \max \left( \frac{\sqrt{k}}{\nu y}, \frac{500 \nu}{y^2 \omega}, \frac{4 \sigma_2 k}{CD^+_{k,2}} \right), 10 \right) \right] \right\}^4 \right\}
\]

tmp<volScalarField> kOmegaSST::F1(const volScalarField& CDkOmega) const
{
    volScalarField CDkOmegaPlus = max
    (   CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );
    volScalarField arg1 = min
    (    min
        (     max
            (     (scalar(1)/betaStar_)*sqrt(k_)/(omega_*y_),
                scalar(500)*nu()/(sqr(y_)*omega_)
            ),
            (4*alphaOmega2_)*k_/(CDkOmegaPlus*sqr(y_))
        ),
        scalar(10)
    );
    return tanh(pow4(arg1));
}

\(F_1 = 0\) in the freestream (\(k - \varepsilon\) model) and \(F_1 = 1\) in the boundary layer (\(k - \omega\) model)
The effective diffusivity for $k$, $(D_{k\text{Eff}}(F_1))$, kOmegaSST.H:

```cpp
tmp<volScalarField> DkEff(const volScalarField& F1) const
{
    return tmp<volScalarField>(
        new volScalarField("DkEff", alphaK(F1)*nut_ + nu())
    );
}
```

Blend $\text{alphaK}_1$ and $\text{alphaK}_2$ using blend function F1, kOmegaSST.H:

```cpp
tmp<volScalarField> alphaK(
    const volScalarField& F1
) const
{
    return blend(F1, alphaK1_, alphaK2_);
}
```

In kOmegaSST.C:

```cpp
alphaK1_(dimensioned<scalar>::lookupOrAddToDict("alphaK1", coeffDict_, 0.85034))
alphaK2_(dimensioned<scalar>::lookupOrAddToDict("alphaK2", coeffDict_, 1.0))
```
\( \frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[ (\nu + \sigma_w \nu_t) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F_1)\sigma_w^2 \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \)

In kOmegaSST.C:

```cpp
tmp<fvScalarMatrix> omegaEqn
  (    
    fvm::ddt(omega_) 
  + fvm::div(phi_, omega_) 
  - fvm::Sp(fvc::div(phi_), omega_) 
  - fvm::laplacian(DomegaEff(F1), omega_) 
  == 
    gamma(F1)*2*S2 
  - fvm::Sp(beta(F1)*omega_, omega_) 
  - fvm::SuSp
      (       
        (F1 - scalar(1))*CDkOmega/omega_, 
        omega_       
      )
    )
);```

\( k - \omega \) SST in OpenFOAM-1.6, Specific dissipation rate eq.
Modify our mykOmegaSST model into kOmegaSSTF


- Upper limit ($\Delta_f$ or $l_t$ below) to the modelled length scale ($L_t$ or $Lt$ below), applied to $\nu_t$:

  $$
  \Delta_f = \alpha \max \left\{ \left| \vec{U} \right| \delta t, \Delta^{1/3} \right\}, \quad \alpha = 3 \quad (\alpha > 1), \quad \hat{\nu}_t = \left( \frac{\min(L_t, \Delta_f)}{L_t} \right)^{4/3} \frac{k}{\omega}
  $$

**kOmegaSST:**

// Re-calculate viscosity
nut_ = a1_*k_/max(a1_*omega_, F2()*sqrt(S2));

**kOmegaSSTF:** (implementation can be improved)

// Compute Filter
scalar alph = 3.0; //Should be in a dictionary
scalarField Lt = sqrt(k_)/(betaStar_*omega_);
scalarField lt = alph*Foam::max(Foam::pow(mesh_.V().field(), 1.0/3.0),
  (mag(U_)*runTime_.deltaT())->internalField());

// Re-calculate viscosity
nut_.internalField() = Foam::min(Foam::pow(lt/Lt,4.0/3.0), 1.0)*
  (a1_*k_/max(a1_*omega_, F2()*sqrt(S2)))->internalField();
Modify our mykOmegaSST model into kOmegaSSTF

```
cd $WM_PROJECT_USER_DIR/src/turbulenceModels/incompressible/RAS/mykOmegaSST/

Find in mykOmegaSST.C the lines saying:

// Re-calculate viscosity
nut_ = a1_*k_/max(a1_*omega_, F2()*sqrt(S2));

Exchange those lines with:

// Compute Filter
scalar alph = 3.0; //Should be in a dictionary
scalarField Lt = sqrt(k_)/(betaStar_*omega_);
scalarField lt = alph*Foam::max(Foam::pow(mesh_.V().field(), 1.0/3.0),
   (mag(U_)*runTime_.deltaT())->internalField());

// Re-calculate viscosity
nut_.internalField() = Foam::min(Foam::pow(lt/Lt,4.0/3.0), 1.0)*
   (a1_*k_/max(a1_*omega_, F2()*sqrt(S2)))->internalField();

Compile with cd ..; wmake libso
```
Modify the pitzDaily case for pimpleFoam

Make sure that you are in the pitzDaily case, and delete the previous results:

```bash
run ; cd pitzDaily ; rm -r [1-9]*
```

Modify the files in `system`, for use with pimpleFoam:

```bash
cp $FOAM_TUTORIALS/incompressible/pimpleFoam/TJunction/system/{fvSolution,fvSchemes} system
sed -i s/epsilon/omega/g system/fvSchemes
sed -i s/epsilon/omega/g system/fvSolution
sed -i s/simpleFoam/pimpleFoam/g system/controlDict
sed -i s/1000/0.3/g system/controlDict
sed -i s/"1;"/"0.0001;"/g system/controlDict
sed -i s/uncompressed/compressed/g system/controlDict
```

Add to `system/controlDict`:

```bash
adjustTimeStep no;
maxCo 5;
pimpleFoam needs one more dictionary:
```

```bash
cp $FOAM_TUTORIALS/incompressible/pimpleFoam/TJunction/constant/turbulenceProperties constant
```

We can re-use the same 0 directory that we modified before.

Make sure that you still specify `mykOmegaSST` in `constant/RASproperties`

Run the case with `pimpleFoam -noFunctionObjects` and make a nice movie of the results.
kOmegaSSTF

The kOmegaSSTF turbulence model is available for OpenFOAM-1.5 at OpenFOAM-extend:


There is a pitzDaily tutorial for the turbFoam solver (no longer in OpenFOAM-1.6 and newer versions), and a utility for viewing the filter function.

It is also used in the Dellenback Abrupt Expansion case-study, which is described in the Turbulence Working Group Wiki:

http://openfoamwiki.net/index.php/Sig_Turbulence_/Dellenback_Abrupt_Expansion