Changes and Settings for Standard Turbulence Model Implementation in OpenFOAM

Author: Sebastian Gomez
Contact: Dr. Svetlana V. Poroseva (poroseva@unm.edu)
Mechanical Engineering, University of New Mexico

The first part of this document will describe the changes made to OpenFOAM's incompressible RAS turbulence models to generate "standard" turbulence model definitions found on NASA's Turbulence Modeling Resource website (TMR). The turbulence models that will be described are Spalart-Allmaras (SA), Shear Stress Transport (SST), and Wilcox's 2006 version of k-ω (k-ω). The standard equations for each turbulence model have been included in the Appendix. The second part of this document will provide the initial conditions and settings used in OpenFOAM to obtain results that were in agreement with NASA for the 2D Zero Pressure Gradient Flat Plate and 2D Bump-in-channel verification cases.

PART 1

Spalart-Allmaras

Changes to the `SpalartAllmaras` model in OpenFOAM were the following:

1. Modified OpenFOAM's definition of the f_{v2} function to be in accordance with NASA's.
2. Eliminated f_{v3} function and c_{v2} coefficient found in OpenFOAM.
3. Added f_{t2}, c_{t3}, and c_{t4} as described in the SA section of the TMR website.
4. Modified $\tilde{S}$ definition to take into account changes listed above.
5. Modified $\tilde{\nu}$ transport equation to take into account changes listed above.

Menter Shear Stress Transport

Changes to the `kOmegaSST` model in OpenFOAM were the following:

1. Modified OpenFOAM's definition of the $CD_{k\omega}$, $arg1$ and $arg2$ functions to be in accordance with NASA's.
2. Eliminated F_3 and F_{23}functions found in OpenFOAM.
3. Modified ν_t definition to be in accordance with NASA's.
4. Modified $\frac{\gamma_\omega}{\nu_t} P$ term and the sign of the last term in the ω transport equation to be in accordance with NASA.
5. Substituted P in k transport equation with P^* as suggested in the TMR website.

Wilcox's 2006 version of *k-ω*

Changes to the `kOmega` model in OpenFOAM were the following:

1. Modified β definition to make it a function instead of a constant.
2. Added f_β, χ_ω and σ_d definitions as listed on the TMR website.
3. Modified ν_t definition to include $\widetilde{\omega}$ as defined by NASA.
4. Added $CD_{k\omega}$ as the last term in the ω transport equation.

Detailed Description

The changes described in this part of the document may be out of order due to the fact that the entire source code isn't included in the text. The user may do a side-by-side comparison between OpenFOAM's default source code and the modified source code to determine exactly where each change was made within the source code.

The changes described in this section correspond to what was done to the .C source files. The .H header files must also be modified to reflect the changes described below. Adding and/or removing constants and functions from the .H files is trivial so those changes will not be included in this section. The variables that use ν_t as part of their definition in the .H files must be changed if the definition of ν_t is modified in the .C file. Some of the model coefficient values in OpenFOAM were slightly different from what was published on the TMR website. The user should make sure that the numerical values for the model coefficients found on the TMR website correspond to those used in OpenFOAM.

Spalart-Allmaras

1. The f_{v2} modification consisted of replacing OpenFOAM's definition with NASA's.

Original OpenFOAM definition:

```
tmp<volScalarField> MySpalartAllmaras::fv2
(
    const volScalarField& chi,
    const volScalarField& fv1
) const
{
    return 1.0/pow3(scalar(1) + chi/Cv2_);
}
```

Modified definition:

```
tmp<volScalarField> MySpalartAllmaras::fv2
(
    const volScalarField& chi,
    const volScalarField& fv1
) const
{
    return 1.0 - chi/(1.0+chi*fv1);
}
```

Spalart-Allmaras

2. To eliminate the f_{v3} function and c_{v2} coefficient found in OpenFOAM**,** the following segments of code were commented/removed from the source code:

```
tmp<volScalarField> MySpalartAllmaras::fv3
(
    const volScalarField& chi,
    const volScalarField& fv1
) const
{
    const volScalarField chiByCv2((1/Cv2_)*chi);

    return
        (scalar(1) + chi*fv1)
       *(1/Cv2_)
       *(3*(scalar(1) + chiByCv2) + sqr(chiByCv2))
       /pow3(scalar(1) + chiByCv2);
}


Cv2_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cv2",
            coeffDict_,
            5.0
        )
    )


Cv2_.readIfPresent(coeffDict());
```

Spalart-Allmaras

3. To add f_{t2}, c_{t3}, and c_{t4} as described in the SA section of the TMR website, the following definitions were added:

```
tmp<volScalarField> MySpalartAllmaras::ft2(const volScalarField&
chi) const
{
    const volScalarField chi2(pow(chi,2));
    return Ct3_*exp(-1.0*Ct4_*chi2);
}


Ct3_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Ct3",
            coeffDict_,
            1.2
        )
    ),
Ct4_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Ct4",
            coeffDict_,
            0.5
        )
    )

Ct3_.readIfPresent(coeffDict());
Ct4_.readIfPresent(coeffDict());
```

Spalart-Allmaras

4. The $\tilde{S}$ definition was modified to take changes listed above into account:

Original OpenFOAM definition:

```
    const volScalarField Stilda
    (
        fv3(chi, fv1)*::sqrt(2.0)*mag(skew(fvc::grad(U_)))
      + fv2(chi, fv1)*nuTilda_/sqr(kappa_*d_)
    );
```

Modified definition:

```
  const volScalarField Stilda
    (
        sqrt(2.0)*mag(skew(fvc::grad(U_)))
      + fv2(chi, fv1)*nuTilda_/sqr(kappa_*d_)
    );
```

5. Modified $\tilde{\nu}$ transport equation to take into account changes listed above.

Original OpenFOAM definition:

```
    tmp<fvScalarMatrix> nuTildaEqn
    (
        fvm::ddt(nuTilda_)
      + fvm::div(phi_, nuTilda_)
      - fvm::laplacian(DnuTildaEff(), nuTilda_)
      - Cb2_/sigmaNut_*magSqr(fvc::grad(nuTilda_))
     ==
        Cb1_*Stilda*nuTilda_
      - fvm::Sp(Cw1_*fw(Stilda)*nuTilda_/sqr(d_), nuTilda_)
    );
```

Modified definition:

```
    tmp<fvScalarMatrix> nuTildaEqn
    (
        fvm::ddt(nuTilda_)
      + fvm::div(phi_, nuTilda_)
      - fvm::laplacian(DnuTildaEff(), nuTilda_)
      - Cb2_/sigmaNut_*magSqr(fvc::grad(nuTilda_))
     ==
        Cb1_*(1.0-ft2(chi))*Stilda*nuTilda_
      -  fvm::Sp((Cw1_*fw(Stilda)*nuTilda_
      -  Cb1_*ft2(chi)*nuTilda_/sqr(kappa_))/sqr(d_), nuTilda_)
    );
```

Menter Shear Stress Transport

1. To modify OpenFOAM's definition of the $CD_{k\omega}$, $arg1$ and $arg2$ functions to be in accordance with NASA's, the following changes were made:

Original OpenFOAM definition for $CD_{k\omega}$:

```
tmp<volScalarField> CDkOmegaPlus = max
(
    CDkOmega,
dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
);
```

Modified definition for $CD_{k\omega}$:

```
tmp<volScalarField> CDkOmegaPlus = max
(
    CDkOmega,
 dimensionedScalar("1.0e-21", dimless/sqr(dimTime), 1.0e-21)
);
```

Original OpenFOAM definition for $arg1$:

```
tmp<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));
```

Modified definition for $arg1$:

```
tmp<volScalarField> arg1 = min
(
        max
        (
            (scalar(1)/betaStar_)*sqrt(k_)/(omega_*y_),
            scalar(500)*nu()/(sqr(y_)*omega_)
        ),
        (4*alphaOmega2_)*k_/(CDkOmegaPlus*sqr(y_))
);

return tanh(pow4(arg1));
```

Original OpenFOAM definition for $arg2$:

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

    return tanh(sqr(arg2));
}
```

Modified definition for $arg2$:

```
tmp<volScalarField> MySSTStd::F2() const
{
    tmp<volScalarField> arg2 =
        max
        (
            (scalar(2)/betaStar_)*sqrt(k_)/(omega_*y_),
            scalar(500)*nu()/(sqr(y_)*omega_)

    );

    return tanh(sqr(arg2));
}
```

Menter Shear Stress Transport

2. Eliminated F_3 and F_{23}functions found in OpenFOAM by removing/commenting the following:

```
tmp<volScalarField> kOmegaSST::F3() const
{
    tmp<volScalarField> arg3 = min
    (
        150*nu()/(omega_*sqr(y_)),
        scalar(10)
    );

    return 1 - tanh(pow4(arg3));
}


tmp<volScalarField> kOmegaSST::F23() const
{
    tmp<volScalarField> f23(F2());

    if (F3_)
    {
        f23() *= F3();
    }

    return f23;
}

F3_
    (
        Switch::lookupOrAddToDict
        (
            "F3",
            coeffDict_,
            false
        )
    )

F3_.readIfPresent("F3", coeffDict());
```

Menter Shear Stress Transport

3. Modified ν_t definition to be in accordance with NASA's:

Original OpenFOAM $\nu_t$definition:

```
    nut_ =
    (
        a1_*k_
      / max
        (
            a1_*omega_,
            b1_*F23()*sqrt(2.0)*mag(symm(fvc::grad(U_)))
        )
    );
```

Modified ν_t definition:

```
    nut_ =
    (
        a1_*k_
      / max
        (
            a1_*omega_,
            b1_*F2()*sqrt(2.0)*mag(skew(fvc::grad(U_)))
        )
    );
```

Menter Shear Stress Transport

4. Modified $\frac{\gamma_\omega}{\nu_t} P$ term and the sign of the last term in the ω transport equation to be in accordance with NASA:

Original OpenFOAM ω transport equation:

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

Modified ω transport equation:

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

Menter Shear Stress Transport

5. Substituted P in k transport equation with P^* as suggested in the TMR website. Defined P^* in OpenFOAM as:

```
volScalarField G2(type() + ".G",
min(G,scalar(20.0)*betaStar_*omega_*k_));
```

Original OpenFOAM k transport equation:

```
    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(k_)
      + fvm::div(phi_, k_)
      - fvm::laplacian(DkEff(F1), k_)
     ==
        G
      - fvm::Sp(betaStar_*omega_, k_)
    );
```

Modified k transport equation:

```
    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(k_)
      + fvm::div(phi_, k_)
      - fvm::laplacian(DkEff(F1), k_)
     ==
        G2
      - fvm::Sp(betaStar_*omega_, k_)
    );
```

Wilcox's 2006 version of *k-ω*

1. To modify β definition to make it a function instead of a constant, the following was removed/commented:

```
    beta_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "beta",
            coeffDict_,
            0.072
        )
    )

beta_.readIfPresent(coeffDict());
```

The following was also added:

```
   beta_
    (
        IOobject
        (
            "beta",
            runTime_.timeName(),
            mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        mesh_,
        dimless
    )

beta_ = 0.0708*fBeta_;
```

Wilcox's 2006 version of *k-ω*

2. Added f_β, χ_ω and σ_d definitions as defined in the TMR website by first adding the following declarations:

```
    fBeta_
     (
         IOobject
         (
             "fBeta",
             runTime_.timeName(),
             mesh_,
             IOobject::NO_READ,
             IOobject::NO_WRITE
         ),
         mesh_,
         dimless
     ),
    Chi_
     (
         IOobject
         (
             "Chi",
             runTime_.timeName(),
             mesh_,
             IOobject::NO_READ,
             IOobject::NO_WRITE
         ),
         mesh_, dimless
     ),
    absChi_
     (
         IOobject
         (
             "absChi",
             runTime_.timeName(),
             mesh_,
             IOobject::NO_READ,
             IOobject::NO_WRITE
         ),
         mesh_, dimless
     ),
alphad_
     (
         IOobject
         (
             "alphad",
             runTime_.timeName(),
             mesh_,
             IOobject::NO_READ,
             IOobject::NO_WRITE
         ),
```

```
        mesh_, dimensionedScalar("zero", dimless, 0.125)
    )
```

Second, the following definitions were added to decrease computational time:

```
    volTensorField GradU(fvc::grad(U_));
    volSymmTensorField Sij(symm(GradU));
    volTensorField Omij(-skew(GradU));
    volScalarField StressLim(Clim_*sqrt(2.0/Cmu_)*mag(Sij));
    volSymmTensorField tauij(2.0*nut_*Sij-((2.0/3.0)*I)*k_);
    volVectorField Gradk(fvc::grad(k_));
    volVectorField Gradomega(fvc::grad(omega_));
```

The production definition term was modified to include changes listed above:

```
volScalarField G(type() + ".G", tauij && GradU);
```

Third, in order to implement the σ_d condition, the following was defined:

```
volScalarField alphadCheck_(Gradk & Gradomega);  //condition to
change alphad_
forAll(alphad_,celli)
{
    if (alphadCheck_[celli] <= 0.0001)
    {
    alphad_[celli]=scalar(0.0);
    }else
    {
    alphad_[celli]=scalar(0.125);
    }
}
```

Fourth, f_β and χ_ω were defined in the following way:

```
Chi_ = (Omij & Omij) && Sij /pow((Cmu_*omega_),3);
absChi_ = mag(Chi_);
fBeta_ = (1.0+85.0*absChi_)/(1.0+100.0*absChi_);  //This term
should be set = to 1 for 2-D
```

*****It should be noted that to model 2-D flows, Pope's correction, denoted by χ_ω, should be set equal to zero.**

Wilcox's 2006 version of *k-ω*

3. Modified ν_t definition to include $\tilde{\omega}$ as defined by NASA.

Original OpenFOAM ν_t definition:

```
nut_ = k_/omega_;
```

Modified ν_t definition:

```
nut_ = k_/ max(omega_,
Clim_*sqrt(2.0/0.09*magSqr(symm(fvc::grad(U_)))));;
```

4. Added $CD_{k\omega}$ as the last term in the ω transport equation by defining it first:

```
volScalarField CDkOmega(alphad_/omega_*(Gradk & Gradomega));
```

Modified the ω transport equation second to add $CD_{k\omega}$:

```
    tmp<fvScalarMatrix> omegaEqn
    (
        fvm::ddt(omega_)
      + fvm::div(phi_, omega_)
      - fvm::laplacian(DomegaEff(), omega_)
     ==
        alpha_*G*omega_/k_
      - fvm::Sp(beta_*omega_, omega_)
      + CDkOmega //Crossflow diffusion term to match 2006
    );
```

PART 2

This section will discuss the following:

i) Values and settings for initial/boundary conditions (i.e.: the 0 folder)

ii) Settings for discretization and solvers (i.e.: fvSchemes and fvSolution)

Values and settings for initial/boundary conditions

2D Zero Pressure Gradient Flat Plate

Spalart-Allmaras

The following values were obtained from the relationships found on the TMR website:

$$\tilde{\nu} = 3\nu = 4.16\times10^{-5}\ \mathrm{m^2/s}$$

$$\nu_t = \tilde{\nu} f_{v1} = \tilde{\nu}\frac{\chi^3}{\chi^3 + c_{v1}^3} = 2.92\times10^{-6}\ \mathrm{m^2/s}$$

$$\chi = \frac{\tilde{\nu}}{\nu}$$

Boundary	**nut** (m^2/s)	**nuTilda** (m^2/s)	**p** (m^2/s^2)	**U** (m/s)
Inlet	calculated 2.92e-6	fixedValue 4.16e-5	zeroGradient -	fixedValue (69.26 0 0)
Upstream	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Plate	fixedValue 0	fixedValue 0	zeroGradient -	fixedValue (0 0 0)
Outlet	zeroGradient -	zeroGradient -	fixedValue 101325	zeroGradient -
Top	zeroGradient -	zeroGradient -	zeroGradient -	zeroGradient -
Front	empty -	empty -	empty -	empty -
Back	empty -	empty -	empty -	empty -
Internal Field	uniform 2.92e-6	uniform 4.16e-5	uniform 101325	uniform 4.1556e-5

SST and *k-ω*

The following values were obtained from the relationships found on the TMR website:

$$k_{farfield} = 9\times10^{-9}a^2 = 1.08\times10^{-3}\,\frac{\mathrm{m}^2}{\mathrm{s}^2}$$

$$\omega_{farfield} = 1\times10^{-6}\,\frac{\rho\,a^2}{\mu} = 8657.5\ \mathrm{s}^{-1}$$

$$\omega_{wall} = 10\,\frac{6\nu}{\beta_1(\Delta d_1)^2} = 4.43\times10^{10}\ \mathrm{s}^{-1}$$

$$\frac{\mu_T}{\mu_\infty} = 0.009 \rightarrow \nu_t = 1.25\times10^{-7}$$

Boundary	**nut** (m^2/s)	**k** (m^2/s^2)	**ω** (m^2/s^2)	**p** (m^2/s^2)	**U** (m/s)
Inlet	calculated 1.25e-7	fixedValue 1.079e-3	fixedValue 8.6575e3	zeroGradient -	fixedValue (69.26 0 0)
Upstream	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Plate	fixedValue 1e-50	fixedValue 1e-50	fixedValue 4.43e10	zeroGradient -	fixedValue (0 0 0)
Outlet	zeroGradient -	zeroGradient -	zeroGradient -	fixedValue 101325	zeroGradient -
Top	zeroGradient -	zeroGradient -	zeroGradient -	zeroGradient -	zeroGradient -
Front	empty -	empty -	empty -	empty -	empty -
Back	empty -	empty -	empty -	empty -	empty -
Internal Field	uniform 1.25e-7	uniform 1.079e-3	uniform 8.6575e3	uniform 101325	uniform (69.26 0 0)

2D Bump-in-channel

Spalart-Allmaras

The variable values were calculated from the same relationships:

$$\tilde{\nu} = 3\nu = 6.93\times10^{-5}\ \mathrm{m^2/s}$$

$$\nu_t = \tilde{\nu} f_{v1} = \tilde{\nu}\frac{\chi^3}{\chi^3 + c_{v1}^3} = 4.86\times10^{-6}\ \mathrm{m^2/s}$$

$$\chi = \frac{\tilde{\nu}}{\nu}$$

Boundary	**nut** (m^2/s)	**nuTilda** (m^2/s)	**p** (m^2/s^2)	**U** (m/s)
Inlet	calculated 4.86e-6	fixedValue 6.926e-5	zeroGradient -	fixedValue (69.26 0 0)
Upstream	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Bump	nutUSpaldingWallFunction 0	fixedValue 0	zeroGradient -	fixedValue (0 0 0)
Downstream	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Outlet	zeroGradient -	zeroGradient -	fixedValue 101325	zeroGradient -
Top	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Front	empty -	empty -	empty -	empty -
Back	empty -	empty -	empty -	empty -
Internal Field	uniform 4.86e-6	uniform 6.926e-5	uniform 101325	uniform (69.26 0 0)

SST and *k-ω*

The relationships to obtain the values used for these simulations were:

$$k_{farfield} = 9\times10^{-9}a^2 = 1.08\times10^{-3}\,\frac{\mathrm{m}^2}{\mathrm{s}^2}$$

$$\omega_{farfield} = 1\times10^{-6}\,\frac{\rho\,a^2}{\mu} = 5220.8\ \mathrm{s}^{-1}$$

$$\omega_{wall} = 10\,\frac{6\nu}{\beta_1(\Delta d_1)^2} = 7.39\times10^{10}\ \mathrm{s}^{-1}$$

$$\frac{\mu_T}{\mu_\infty} = 0.009 \rightarrow \nu_t = 2.08\times10^{-7}$$

Boundary	**nut** (m^2/s)	**k** (m^2/s^2)	**ω** (m^2/s^2)	**p** (m^2/s^2)	**U** (m/s)
Inlet	calculated 2.078e-7	fixedValue 1.079e-3	fixedValue 5.2208e3	zeroGradient -	fixedValue (69.26 0 0)
Upstream	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Bump	nutUSpaldingWallFunction 1e-50	fixedValue 1e-50	fixedValue 7.389e10	zeroGradient -	fixedValue (0 0 0)
Upstream	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Outlet	zeroGradient -	zeroGradient -	zeroGradient -	fixedValue 101325	zeroGradient -
Top	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -	symmetryPlane -
Front	empty -	empty -	empty -	empty -	empty -
Back	empty -	empty -	empty -	empty -	empty -
Internal Field	uniform 2.078e-7	uniform 1.079e-3	uniform 5.2208e3	uniform 101325	uniform (69.26 0 0)

Settings for discretization and solvers

The files shown correspond to the simulation run on the finest mesh for each flow case. Relaxation factors may be increased for coarser meshes. The OpenFOAM header for has been excluded.

2D Zero Pressure Gradient Flat Plate

fvSchemes

```
ddtSchemes
{
    default         steadyState;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,epsilon) bounded Gauss upwind;
    div(phi,k)      bounded Gauss upwind;
    div(phi,R)      bounded Gauss upwind;
    div(phi,nuTilda) bounded Gauss upwind;
    div(R)          Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div(DomegaEff,omega) bounded Gauss upwind;
    div(phi,omega) bounded Gauss upwind;

}
laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
    laplacian(DomegaEff,omega) Gauss linear corrected;
    laplacian(phi,omega) Gauss linear corrected;
}
interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p               ;
}
```

fvSolution

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-20;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    epsilon
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    R
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    nuTilda
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }
    omega
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }
}
```

```
SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        U               0.7;
        k               0.7;
        epsilon         0.7;
        R               0.7;
        nuTilda         0.7;
        omega           0.7;
    }
}
```

2D Bump-in-channel

fvSchemes

```
ddtSchemes
{
    default         steadyState;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,epsilon)  bounded Gauss upwind;
    div(phi,omega)  bounded Gauss upwind;
    div(phi,k)      bounded Gauss upwind;
    div(phi,nuTilda) bounded Gauss upwind;
    div(phi,R)      bounded Gauss upwind;
    div(R)          Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div(DomegaEff,omega) bounded Gauss upwind;
}

laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
    laplacian(DomegaEff,omega) Gauss linear corrected;
    laplacian(phi,omega) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}
```

fvSolution

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  FDIC;
        tolerance       1e-15;
        relTol          0.01;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    epsilon
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    R
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }

    nuTilda
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }
    omega
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-20;
        relTol          0;
    }
}
```

```
SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    fields
    {
        p               0.1;
    }
    equations
    {
        U               0.2;
        k               0.2;
        epsilon         0.2;
        R               0.2;
        nuTilda         0.2;
        omega           0.2;
    }
}
```

APPENDIX

Standard Turbulence Model Equations

Spalart-Allmaras

$$\frac{\partial \tilde{\nu}}{\partial t} + \bar{u}_j \frac{\partial \tilde{\nu}}{\partial x_j} = c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} - \left[c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left((\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j}\right) + c_{b2}\frac{\partial \tilde{\nu}}{\partial x_i}\frac{\partial \tilde{\nu}}{\partial x_i}\right].$$

The closure functions are defined as

$$f_{t2} = c_{t3} \exp(-c_{t4}\chi^2)$$

$$\chi = \frac{\tilde{\nu}}{\nu}$$

$$\tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}$$

$$\Omega = \sqrt{2W_{ij}W_{ij}}$$

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

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}$$

$$c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}$$

$$f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6}$$

$$g = r + c_{w2}(r^6 - r)$$

$$r = \min\left[\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right]$$

where *d* is the distance from the field point to the nearest wall.

The eddy viscosity is computed from

$$\nu_t = \tilde{\nu} f_{v1}.$$

The values for the model coefficients can be found in Table 1.

c_{b1}	c_{b2}	κ	σ	c_{t3}	c_{t4}	c_{w2}	c_{w3}
0.1355	0.622	0.41	$\frac{2}{3}$	1.2	0.5	0.3	2

Table 1: Spalart-Allmaras model coefficients

Menter Shear Stress Transport

$$\frac{\partial k}{\partial t} + \bar{u}_j \frac{\partial k}{\partial x_j} = P^* - \beta^* \omega k + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_k \nu_t)\frac{\partial k}{\partial x_j}\right].$$

Transport of the specific turbulence dissipation rate is described by

$$\frac{\partial \omega}{\partial t} + \bar{u}_j \frac{\partial \omega}{\partial x_j} = \frac{\gamma_\omega}{\nu_t} P - \beta_\omega \omega^2 + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_\omega \nu_t)\frac{\partial \omega}{\partial x_j}\right] + 2(1 - F_1)\frac{\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j},$$

with model functions defined as

$$P^* = \min(P, 20\beta^* \omega k)$$

$$P = \tau_{ij}\frac{\partial \bar{u}_i}{\partial x_j}$$

$$\tau_{ij} = 2\nu_t S_{ij} - \frac{2}{3}k\delta_{ij}$$

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

$$\nu_t = \frac{a_1 k}{\max(a_1 \omega, \Omega F_2)}$$

$$\Omega = \sqrt{2W_{ij}W_{ij}}$$

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

$$F_2 = \tanh(arg_2^2)$$

$$arg_2 = \max\left(2\frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\nu}{d^2 \omega}\right)$$

$$F_1 = \tanh(arg_1^4)$$

$$arg_1 = \min\left[\max\left(2\frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\nu}{d^2 \omega}\right), \frac{4\sigma_{\omega 2} k}{CD_{k\omega} d^2}\right]$$

$$CD_{k\omega} = \max\left(2\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-20}\right)$$

where the variable d is the distance from the field point to the nearest wall. The constants in the transport equations that don't have a number as part of the subscript are obtained through a blending function of the following form:

$$\phi = F_1\phi_1 + (1 - F_1)\phi_2$$

where ϕ is the value of the constant without a number in the subscript and ϕ_1 and ϕ_2 represent constants 1 and 2. For example, σ_k is defined as

$$\sigma_k = F_1\sigma_{k1} + (1 - F_1)\sigma_{k2}$$

Values for the constant coefficients can be found in Table 2 and the remaining model coefficients are defined as

$$\gamma_1 = \frac{\beta_1}{\beta^*} - \frac{\sigma_{\omega 1}\kappa^2}{\sqrt{\beta^*}} \text{ and } \gamma_2 = \frac{\beta_2}{\beta^*} - \frac{\sigma_{\omega 2}\kappa^2}{\sqrt{\beta^*}} \ .$$

$\boldsymbol{\sigma_{k1}}$	$\boldsymbol{\sigma_{k2}}$	$\boldsymbol{\sigma_{\omega 1}}$	$\boldsymbol{\sigma_{\omega 2}}$	$\boldsymbol{\beta_1}$	$\boldsymbol{\beta_2}$	$\boldsymbol{\beta^*}$	$\boldsymbol{\kappa}$	$\boldsymbol{a_1}$
0.85	1	0.5	0.856	0.075	0.0828	0.09	0.41	0.31

Table 2: Menter SST model coefficients

Wilcox's 2006 version of *k-ω*

$$\frac{\partial k}{\partial t} + \bar{u}_j \frac{\partial k}{\partial x_j} = P - \beta^* \omega k + \frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_k \frac{k}{\omega}\right)\frac{\partial k}{\partial x_j}\right]$$

$$\frac{\partial \omega}{\partial t} + \bar{u}_j \frac{\partial \omega}{\partial x_j} = \frac{\gamma \omega}{k} P - \beta \omega^2 + \frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_\omega \frac{k}{\omega}\right)\frac{\partial \omega}{\partial x_j}\right] + \frac{\sigma_d}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j},$$

where

$$P = \tau_{ij}\frac{\partial \bar{u}_i}{\partial x_j}$$

$$\tau_{ij} = 2\nu_t S_{ij} - \frac{2}{3}k\delta_{ij}$$

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

$$\nu_t = \frac{k}{\tilde{\omega}}$$

$$\tilde{\omega} = \max\left[\omega, C_{\text{lim}}\sqrt{\frac{2S_{ij}S_{ij}}{\beta^*}}\right]$$

The constant model coefficient values can be found in Table 3. Additional relationships are defined as

$$f_\beta = \frac{1 + 85\chi_\omega}{1 + 100\chi_\omega}, \qquad \chi_\omega = \left|\frac{\Omega_{ij}\Omega_{jk}S_{ki}}{(\beta^*\omega)^3}\right|, \qquad \Omega_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} - \frac{\partial \bar{u}_j}{\partial x_i}\right).$$

$$\sigma_d = \begin{cases} 0, \frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \le 0 \\ \frac{1}{8}, \frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} > 0 \end{cases}$$

σ_k	σ_ω	β^*	γ	C_{lim}	β	β_0
0.6	0.5	0.09	$\frac{13}{25}$	$\frac{7}{8}$	$\beta_0 f_\beta$	0.0708

Table 3: *k-ω* model coefficients