OpenFOAM $k - \omega - SST$ Notes

Antti Mikkonen, a.mikkonen@iki.fi
Tampere University of Technology

January 28, 2016

Contents

1 TODO

CHECK BOUNDARY G!

The G given to omega boundary condition is $G = \frac{P_{dev}}{\rho}$. See below.

2 Disclaimer

This paper was written in the process of my personal studying of turbulence modeling of OpenFOAM and reflects my growing understanding on the subject. **This paper should not be considered as a mature scientific publication or a manual.** I'm publishing this in the hope that after a couple of iterations (maybe by other authors) this paper will mature into something useful. I also couldn't find any up-to-date explanation of OpenFOAM $k - \omega - SST$ code so I made one myself.

3 Introduction

This paper explores the wonders of OpenFOAM turbulence modeling, mostly $k - \omega - SST$. In Section 4 Menter's $k - \omega - SST$ 2003 [2] model is briefly given. In Section 5 OpenFOAM implementation is compared with $k - \omega - SST$ and in subsection 5.1 the functional differences are briefly listed.

Note that OpenFOAM is in constant development and not all details in this paper might not be up-to-date. The version should roughly agree to the OpenFOAM 3.0.1. but git pull from the developer repository is run regularly.

4 Menter SST 2003 [2][3]

Menter's notation used in this section.

k transport equation

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho U_j k)}{\partial x_j} = \tilde{P}_k - \beta^* \rho k \omega + \frac{\partial}{\partial x_i}\left[(\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_i}\right] \tag{1}$$

ω transport equation, note that the typo in [2] is fixed as suggested in [3].

$$\frac{\partial(\rho\omega)}{\partial t} + \frac{\partial(\rho U_i \omega)}{\partial x_i} = \frac{\alpha \tilde{P}_k}{\nu_t} - \beta\rho\omega^2 + \frac{\partial}{\partial x_i}\left[(\mu + \sigma_\omega \mu)\frac{\partial \omega}{\partial x_i}\right] + 2(1-F_1)\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{2}$$

$$\mu_t = \frac{\rho a_1 k}{\max(a_1\omega, SF_2)} \tag{3}$$

$$S = \sqrt{2S_{ij}S_{ij}} \tag{4}$$

$$S_{ij} = \frac{1}{2}\left(\partial_j U_i + \partial_i U_j\right) \tag{5}$$

$$\tilde{P}\text{=min(P,10 }\beta^*\ \rho\omega\text{k)} \tag{6}$$

$$\begin{aligned} P &= \mu_t \partial_j U_i (\partial_j U_i + \partial_i U_j) \\ &= \mu_t grad(U) 2 S_{ij} \end{aligned} \tag{7}$$

$$CD_{k\omega} = \max(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-10}) \tag{8}$$

$$F_1 = \tanh(arg_1^4) \tag{9}$$

$$arg_1 = \min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right), \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right] \tag{10}$$

$$F_2 = tanh(arg_2^2) \tag{11}$$

$$arg_2 = \max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right) \tag{12}$$

The constants are given in a table below. **Note the different names!**

5 Menter SST 2003 and OpenFOAM

OpenFOAM notation used in this section unless other ways mentioned. For example OpenFOAM's CDkOmega is marked as $CD_{k\omega}$.

All the found functional differences are briefly listed in section 5.1 Functional differences. All the parts of the code are studied in section 5.2 Understand OpenFOAM implementation in more detail.

5.1 List of functional differences

- F_1 is different because arg1 and $CD^+_{k\omega}$ are different.
  - arg1 has an extra min function. The minimum for arg1 is 10. See listing 3. The difference in $CD^+_{k\omega}$also affect this. See subsection 5.2.4.
  - $CD^+_{k\omega}$ has a different max function that of Menter's. Both OpenFOAM's $CD^+_{k\omega}$and Menter's $CD_{k\omega}$,Eq. 8, are limited to $1.0e-10$ but OpenFOAM's $CD^+_{k\omega}$does not include density. Menter's $CD_{k\omega}$does include density. See subsection 5.2.4.
- F_2 is different because arg1 is different.
  - $arg_2$is limited to values larger than 100. See subsection 5.2.5
- F_3(not included in Menter's paper at all, see [1]) An extra limiter, maximum value of 10. See subsection 5.2.2.
- Turbulent kinetic energy production is different in both k- an ω-equations. For incompressible flow there is no difference. See subsection 5.2.14 and 5.2.15

5.2 Understanding OpenFoam implementation

Some constants and terms have different names in Menters paper [2] and in OpenFoam. Here is a short description of diffenrences

SST-2003	OpenFoam	Short	Value	Notes
β^*	betaStar_	β^*	0.09	
α_1	gamma1_	γ_1	5/9	α replaced by γ
β_1	beta1_	β_1	0.075	
σ_{k1}	alphaK1_	α_{k1}	0.85	σ replaced by α(0.5 for CFX 15)
$\sigma_{\omega 1}$	alphaOmega1_	$\alpha_{\omega 1}$	0.5	
α_2	gamma2_	γ_2	0.44	
β_2	beta2_	β_2	0.0828	
σ_{k2}	alphaK2_	α_{k2}	1	
$\sigma_{\omega 2}$	alphaOmega2_	$\alpha_{\omega 2}$	0.856	
a_1	a1_	a_1	0.31	
	b1_	b_1	1	Menter used a literal
	c1_	c_1	10	Menter used a literal
	F3_	F_3	false	Rough walls switch [1]
	alpha	α		one fields for single-phase

5.2.1 α (extra term)

(Alpha-field) is not present in Menter's paper. I didn't fully investigate what it actually is but both single-phase compressilbe and incompressible turbulence models are given a geometricOneField in constructors and multi-phase models are given a volScalarField. Therefore I assume it has something to to do with multiphase turbulence modeling and can be safely ignored in single-phase models. All terms except kSource() and omegaSource() are multiplied by alpha-field in k and omega equations. Alpha-field is not present anywhere else in the code.

5.2.2 F_3 (extra term)

(F3) is a switch for additional rough wall modeling proposed by Hellsten in 1998 [1]. F3 is disable by default and has no effect on the result. If enabled F3 multiplies F2. F2 is only present in the code as a part of F23. If F3 is disabled F23 is the same as F2.

5.2.3 $CD_{k\omega}$ (different)

(CDkOmega) **is different from that of Menters.**

```
    volScalarField CDkOmega
    (
        (2*alphaOmega2_)*(fvc::grad(k_) & fvc::grad(
            omega_))/omega_
    );
```

Listing 1: CDKOmega. kOmegaSST.C

Writing listing 1 as an equation

$$CD_{k\omega} = 2\alpha_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{13}$$

Comparing to Menter $CD_{k\omega}$, Eq. 8, density and maximum are missing. $CD_{k\omega}$ is used only in two places in the code. One is definition of $CD^{*}_{k\omega}$used in blending function F_1 ,see next paragraph, and the other is ω-equation

```
          - fvm::SuSp
            (
                alpha*rho*(F1 - scalar(1))*CDkOmega/omega_,
                omega_
            )
```

Listing 2: Use of CDKOmega. kOmegaSST.C

In equation form listing 2 reads as

$$\begin{aligned}
&-\alpha\rho(F_1 - 1)CD_{k\omega}\omega/\omega \\
&= \alpha\rho(1 - F_1)CD_{k\omega} \\
&= \alpha\rho(1 - F_1)2\alpha_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \\
&= \alpha 2(1 - F_1)\rho\alpha_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}
\end{aligned} \tag{14}$$

Comparing Eq. 14 and ω-equation 2 and remembering that α can be ignored and that OpenFoam's $\alpha_{\omega 2}$ is the same as $\sigma_{\omega 2}$in Menter's paper, the equations correspond to each other.

5.2.4 F_1, $CD^{+}_{k\omega}$, arg1 (different)

is the other place in code where $CD_{k\omega}$is used as mentioned in last paragraph. The $F1$ is calculated as

```
template<class BasicTurbulenceModel>
tmp<volScalarField> kOmegaSST<BasicTurbulenceModel>::kOmegaSST::F1
(
    const volScalarField& CDkOmega
) const
{
    tmp<volScalarField> CDkOmegaPlus = max
    (
        CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );

    tmp<volScalarField> arg1 = min
    (
        min
        (
            max
            (
                (scalar(1)/betaStar_)*sqrt(k_)/(omega_*y_),
                scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omega_)
            ),
            (4*alphaOmega2_)*k_/(CDkOmegaPlus*sqr(y_))
        ),
        scalar(10)
    );

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

Listing 3: F1. kOmegaSST.C

Writing $CD^+_{k\omega}$ in equation form

$$\begin{aligned} CD^+_{k\omega} &= \max(CD_{k\omega}, 10^{-10}) \\ &= \max(2\alpha_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}) \end{aligned} \tag{15}$$

which is a limited form of $CD_{k\omega}$ and **does not have density included** as Menter's $CD_{k\omega}$does. **There is a functional difference!**

Writing arg1 in equation form

$$arg1 = \min\left\{\min\left[\max\left(\frac{1}{\beta^*}\frac{\sqrt{k}}{\omega y}, \frac{500\mu}{\rho y^2\omega}\right), \frac{4\alpha_{\omega 2}k}{CD^+_{k\omega}y^2}\right], 10\right\} \tag{16}$$

This form has an **extra min clause setting the maximum value of arg1 as 10**. Also the omission of density from $CD^+_{k\omega}$ affects arg1. **Density is also omitted from the term** $\frac{4\alpha_{\omega 2}k}{CD^+_{k\omega}y^2}$ yielding the correct units. **There is a functional difference!**

The form of F_1in return statement is equivalent to that of Menter's . However, **the differences in arg1 and in $CD^+_{k\omega}$cause a functional difference in the value of F_1!**

5.2.5 F_2 (different)

The code

```
template<class BasicTurbulenceModel>
```

```
70 tmp<volScalarField> kOmegaSST<BasicTurbulenceModel>::kOmegaSST::F2() const
71 {
72     tmp<volScalarField> arg2 = min
73     (
74         max
75         (
76             (scalar(2)/betaStar_)*sqrt(k_)/(omega_*y_),
77             scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omega_)
78         ),
79         scalar(100)
80     );
81
82     return tanh(sqr(arg2));
83 }
```

Listing 4: F2. kOmegaSST.C

Writing F_2as an equation

$$
\begin{aligned}
arg_2 &= \min\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\mu}{\rho y^2\omega}\right), 100\right] \\
F_2 &= \tanh(arg_2^2)
\end{aligned}
\tag{17}
$$

Again there is a functional difference! The **min function is not present in Menter's paper**.

5.2.6 Square strain invariant S^2 (identical)

Essentially the same code for S^2 can be found in two places. One is given here, the other can be found on lines 410-411 of kOmegaSST.C.

Strain invariant $S = \sqrt{2S_{ij}S_{ij}}$ is given in a square form in OpenFOAM

$$
S^2 = 2S_{ij}S_{ij} \tag{18}
$$

```
124 template<class BasicTurbulenceModel>
125 void kOmegaSST<BasicTurbulenceModel>::correctNut()
126 {
127     correctNut(2*magSqr(symm(fvc::grad(this->U_))));
128 }
```

Listing 5: S2. kOmegaSST.C

Square strain invariant is given as a parameter for the function *correctNut* and in equation form reads

$$
\begin{aligned}
S^2 &= 2|symm(\frac{\partial U_i}{\partial x_j})|^2 \\
&= 2|\frac{1}{2}(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})|^2 \\
&= 2\left(\frac{1}{2}(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})\right) : \left(\frac{1}{2}(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})\right) \\
&= 2S_{ij}S_{ij}
\end{aligned}
\tag{19}
$$

5.2.7 Turbulent kinematic viscosity $\nu_t = \frac{\mu_t}{\rho}$ (identical)

In OpenFOAM code the turbulent kinematic viscosity is given as

```
111 template<class BasicTurbulenceModel>
112 void kOmegaSST<BasicTurbulenceModel>::correctNut(const volScalarField& S2)
113 {
114     this->nut_ = a1_*k_/max(a1_*omega_, b1_*F23()*sqrt(S2));
115     this->nut_.correctBoundaryConditions();
116     fv::options::New(this->mesh_).correct(this->nut_);
117 
118     BasicTurbulenceModel::correctNut();
119 }
```

Listing 6: correctNut. kOmegaSST.C

where the line 114 is relevant. In equation form this read

$$
\begin{aligned}
\nu_t &= \frac{a_1 k}{\max(a_1\omega, b_1 F_{23}\sqrt{S^2})} \\
&= \frac{a_1 k}{\max(a_1\omega, b_1 F_{23} S)}
\end{aligned}
\tag{20}
$$

Remembering that on default setting $b_1 = 1$ and $F_{23} = F_2$ this correspond perfectly to Menter's paper.

5.2.8 Effective diffusivity for k, $D_{k,eff}$ (identical)

OpenFOAM uses diffusivity instead of viscosity in Laplacian terms. Note that this function is the only place in the code where α_k is used.

```
        //- Return the effective diffusivity for k
        tmp<volScalarField> DkEff(const volScalarField& F1) const
        {
            return tmp<volScalarField>
            (
                new volScalarField("DkEff", alphaK(F1)*
                    this->nut_ + this->nu())
            );
        }
```

Listing 7: DkEff. kOmegaSST.H

In equation form this reads

$$D_{k,eff} = \alpha_k \nu_t + \nu \tag{21}$$

5.2.9 Effective diffusivity for ω, $D_{\omega,eff}$ (identical)

Note that this function is the only place in the code where α_ω is used.

```
        //- Return the effective diffusivity for omega
        tmp<volScalarField> DomegaEff(const
            volScalarField& F1) const
        {
            return tmp<volScalarField>
            (
                new volScalarField
                (
                    "DomegaEff",
                    alphaOmega(F1)*this->nut_ + this->nu
                        ()
                )
            );
        }
```

Listing 8: DomegaEff. kOmegaSST.H

In equation form this reads

$$D_{\omega,eff} = \alpha_\omega \nu_t + \nu \tag{22}$$

5.2.10 Blend function (identical)

α_k,α_ω, β, and γ are blended between values corresponding to $k-\epsilon$ and $k-\omega$ model according to

$$\psi = F_1(\psi_1 - \psi_2) + \psi_2 \tag{23}$$

where ψ is a the blended coefficient. $F_1 = 0$ far from surface and $k-\epsilon$ values (subscript 2) are chosen [2].

5.2.11 Laplacian terms in k- and ω-equations (identical)

Both k- and ω-equations use effective diffusivity in the Laplacian. The form is equivalent for both equation and k-equation is used as an example here. In equation form

$$\frac{\partial}{\partial x_i}\left(\alpha\rho D_{k,eff}\frac{\partial k}{\partial x_i}\right) = \frac{\partial}{\partial x_i}\left(\alpha\rho(\alpha_k\nu_t+\nu)\frac{\partial k}{\partial x_i}\right) = \frac{\partial}{\partial x_i}\left(\alpha(\alpha_k\mu_t+\mu)\frac{\partial k}{\partial x_i}\right) \tag{24}$$

which is equivalent to Menter's remembering that α is a geometricOneField. **Note that α and α_t have nothing to do with each other!**

Note that with two parameters the OpenFOAM Laplacian is intepreted as

$$\text{laplacian}(Gamma, phi) = \nabla\cdot\Gamma\nabla\phi = \frac{\partial}{\partial x_i}\left(\Gamma\frac{\partial\phi}{\partial x_i}\right) \tag{25}$$

5.2.12 Temporal and convective terms in k- and ω-equations (identical)

are self-explanatory. Just remember that α is a geometricOneField.

Turbulent kinetic energy sink (identical) OpenFOAM code read

```
      - fvm::Sp(alpha*rho*betaStar_*omega_, k_)
```

Listing 9: k sink. kOmegaSST.C

in equation form

$$-\alpha\rho\beta^*\omega k \tag{26}$$

and is equivalent to Menter's.

5.2.13 Turbulent frequency sink (identical)

OpenFOAM code read

```
          - fvm::Sp(alpha*rho*beta*omega_, omega_)
```

Listing 10: omega sink. kOmegaSST.C

in equation form

$$-\alpha\rho\beta\omega\omega = -\alpha\rho\beta\omega^2 \tag{27}$$

and is equivalent to Menter's.

5.2.14 Turbulent kinetic energy production P (different)

The turbulent kinetic energy term in OpenFOAM $k{-}\omega$-SST is a rather confusing one. There is also some confusion in the literature about what is the correct production term for $k-\omega$-SST 2003. In Menters original paper [2], however, the production is given as in Eq. 6 and 7

$$\tilde{P}\text{=min(P,10 } \beta^* \ \rho\omega\text{k)}$$

$$\begin{aligned} P &= \partial_j U_i \tau_{ij}^{turb} \\ \tau_{ij}^{turb} &= \mu_t 2 S_{ij} \\ P &= \mu_t \partial_j U_i 2 S_{ij} \end{aligned}$$

The G production term in OpenFOAM code

```
    volScalarField GbyNu((tgradU() && dev(twoSymm(tgradU()))));
    volScalarField G(this->GName(), nut*GbyNu);
```

Listing 11: Production term. kOmegaSST.C

and production terms in turbulent kinetic energy transport equation

```
        min(alpha*rho*G, (c1_*betaStar_)*alpha*rho*k_*omega_)
      - fvm::SuSp((2.0/3.0)*alpha*rho*divU, k_)
```

Listing 12: k production. kOmegaSST.C

do not correspond to that of Menters [2]. They more closely resemble equations derived from

$$\begin{aligned} P &= \tau_{ij}^{turb} \partial_j U_i \\ \tau_{ij}^{turb} &= -\overline{\rho u_i'' u_j''} \approx 2\mu_t dev(S_{ij}) - \frac{2}{3}\rho k \delta_{ij} \end{aligned} \tag{28}$$

Combining the above

$$\boxed{\begin{aligned} P = \tau_{ij}^{turb} \partial_j U_i &= grad(U) : (2\mu_t dev(S) - \frac{2}{3}\rho k I) \\ &= \underbrace{\mu_t grad(U) : (dev(twoSymm(grad(U))}_{G\rho} \\ &- \frac{2}{3}\rho k div(U) \end{aligned} \qquad (29)}$$

The first term is OpenFOAMs G times ρ. The second term can be solved with a little tensor exercise

$$
\begin{aligned}
-grad(U):(\frac{2}{3}\rho kI) &= -\frac{2}{3}\rho k\begin{pmatrix}\partial_1 U_1 & \partial_1 U_2 & \partial_1 U_3\\ \partial_2 U_2 & \partial_2 U_2 & \partial_2 U_3\\ \partial_3 U_3 & \partial_3 U_2 & \partial_3 U_3\end{pmatrix}:\begin{pmatrix}1 & & \\ & 1 & \\ & & 1\end{pmatrix}\\
&= -\frac{2}{3}\rho k(\partial_1 U_1+\partial_2 U_2+\partial_3 U_3)\\
&= -\frac{2}{3}\rho k div(U)
\end{aligned}
\tag{30}
$$

which is the production term outside the min function in OpenFOAMs turbulent kinetic energy transport equation.

Clearly, **there is a functional difference in the production term as it is delivered from a different source!** Also, the production limiter operates only on the production term arising from the deviatoric part of τ_{ij}^{turb}. For incompressible flow, however, there is no functional difference because the $div(U)$ term becomes zero and the G term arises in equivalent form from Menter production term

Starting from Menter's P and continuing to deviatoric and hydrostatic parts

$$
\begin{aligned}
P &= \mu_t\partial_j U_i(\partial_j U_i+\partial_i U_j)\\
&= \mu_t grad(U):twoSymm(grad(U))\\
&= \mu_t grad(U):(dev(twoSymm(grad(U)))+hyd(twoSymm(grad(U))))\\
&= \mu_t grad(U):(dev(twoSymm(grad(U)))+\frac{1}{3}tr(twoSymm(grad(U)))I)\\
&= \underbrace{\mu_t grad(U):dev(twoSymm(grad(U)))}_{G\rho}+\mu_t grad(U):\frac{1}{3}tr(twoSymm(grad(U)))I
\end{aligned}
\tag{31}
$$

The first part is OpenFOAM $G\rho$ and studying the second term leads to a $div(U)$ term

$$
\begin{aligned}
tr(twoSymm(grad(U))) &= 2\partial_1 U_1+2\partial_2 U_2+2\partial_3 U_3\\
&= 2div(U)\\
\mu_t grad(U):\frac{1}{3}tr(twoSymm(grad(U)))I &= \frac{2}{3}\mu_t\left(div(U)\right)^2
\end{aligned}
$$

5.2.15 ω production term $\frac{\gamma\tilde{P}}{\nu_t}$ (different)

The same turbulent kinetic energy production term as in section 5.2.14 is used for turbulent frequency production also. The relevant lines in omega equation are

```
437                  alpha*rho*gamma
438                 *min
439                  (
440                      GbyNu,
441                      (c1_/a1_)*betaStar_*omega_*max(a1_*omega_
                             , b1_*F23()*sqrt(S2))
442                  )
443               - fvm::SuSp((2.0/3.0)*alpha*rho*gamma*divU,
                    omega_)
```

Listing 13: omega production. kOmegaSST.C

Writing this in equation form

$$\alpha\rho\gamma \min\left(\frac{G}{\nu_t}, \frac{c_1}{a_1}\beta^*\omega \max[a_1\omega, b_1 F_{23}\sqrt{S^2}]\right) = \frac{\alpha\gamma}{\nu_t}\min\left(\rho G, \frac{\rho c_1}{a_1}\nu_t\beta^*\omega \max[a_1\omega, b_1 F_{23} S]\right) \tag{32}$$

Studying the second term further. Remember from Eq.3 $\nu_t = \frac{a_1 k}{\max(a_1\omega, SF_2)}$ and that $F_{23} = F_2$if $F_3$is not used as is the default and $b_1 = 1$ by default.

$$\begin{aligned}
c_1\beta^*\rho\nu_t\omega \max[\omega, \frac{b_1}{a_1}F_{23}S] &= c_1\beta^*\rho\nu_t\omega \max[\omega, \frac{b_1}{a_1}F_{23}S] \\
&= c_1\beta^*\rho\frac{a_1 k}{\max(a_1\omega, SF_{23})}\omega \max[\omega, \frac{b_1}{a_1}F_{23}S] \\
&= c_1\beta^*\rho k\omega\frac{1}{\max(\omega, \frac{SF_{23}}{a_1})}\max[\omega, \frac{SF_{23}}{a_1}] \\
&= c_1\beta^*\rho k\omega
\end{aligned} \tag{33}$$

This limit corresponds to that of Menters. Note, however, that the G only has part of the Menters P as mentioned in previous chapter. The term on line 443 can be derived similarly to that on turbulent kinetic transport equation if we assume $\omega = \frac{k}{\nu_t}$ which is not strictly correct for an SST model

$$\boxed{\begin{aligned}
\frac{\gamma P}{\nu_t} &= \frac{\gamma}{\nu_t}(G\rho - \frac{2}{3}\rho k div(U)) \\
&= \rho\gamma\frac{G}{\nu_t} - \frac{2}{3}\rho\gamma div(U)\frac{k}{\nu_t} \\
&= \rho\gamma\frac{G}{\nu_t} - \frac{2}{3}\rho\gamma div(U)\omega
\end{aligned} \qquad (34)}$$

The outcome is that there is a functional difference similar to that in turbulent kinetic energy production!

5.2.16 Random remarks

Implementation of the div(U) is a little confusing in the code

```
    volScalarField divU(fvc::div(fvc::absolute(this->phi(), U)));
```

Listing 14: divU. kOmegaSST.C

The “absolute” function does nothing for unmoving mesh as can be seen from ../src/finiteVolume/finiteVolume/fvc/fvcMeshPhi.C

```
Foam::tmp<Foam::surfaceScalarField> Foam::fvc::absolute
(
    const tmp<surfaceScalarField>& tphi,
    const volVectorField& U
)
{
    if (tphi().mesh().moving())
    {
        return tphi + fvc::meshPhi(U);
    }
    else
    {
        return tmp<surfaceScalarField>(tphi, true);
    }
}
```

Listing 15: absolute(phi,U). fvcMeshPhi.C

5.3 Mysteries about OpenFoam implementation

What is the α field exactly? What are kSource and omegaSource terms and why are they not multiplied by α like everything else? How does fvOptions and Qsas terms work? Why is the production term written in such a mighty confusing manner? What does the comment in the H-file about (alpha = 1/sigma) mean? I'm guessing this is outdated as there is nothing in the code to explain this. Many turbulence papars on turbulence model use a sigma corresponding to alpha = 1/sigma but not Menters $k - \omega - SST$.

6 Modified turbulent kinetic energy production

Mu mods here.

6.1 Consistent SST 2003

6.2 SST-V

6.3 Kato Launder

6.4 SST-RC

6.5 SST-RC-Hellsten

7 Recommended reading

- OpenFoam ProgrammersGuide. Especially chapeter “1.3.9 Operations exclusive to tensors of rank 2“
- http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2013/DanielLindblad/k-Omega-SST-DES-Report.pdf
- http://turbmodels.larc.nasa.gov/sst.html
- http://cfd.mace.manchester.ac.uk/flomania/pds_papers/file_pds-1068134610Menter-SST-paper.pdf
- http://www.personal.psu.edu/faculty/c/x/cxc11/508/Index_Notation_C.pdf
- http://www.cfd-online.com/Wiki/Kato-Launder_modification

8 Some tensor exercises

8.0.1 Dyadics

A dyadic or dyadic tensor is a second order tensor, written in a notation that fits in with vector algebra.

Commutative

$$A : B = B : A \tag{35}$$

distributive The dyadic product is distributive over vector addition

$$\mathbf{a}(\mathbf{b} + \mathbf{c}) = \mathbf{ab} + \mathbf{ac} \tag{36}$$

$$
\begin{aligned}
A &= \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \\
B &= \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} \\
C &= \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix} \\
A:(B+C) &= \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} : \begin{pmatrix} B_{11}+C_{11} & B_{12}+C_{12} & B_{13}+C_{13} \\ B_{21}+C_{21} & B_{22}+C_{22} & B_{23}+C_{23} \\ B_{31}+C_{31} & B_{32}+C_{32} & B_{33}+C_{33} \end{pmatrix} \\
&= \\
&+ A_{11}(B_{11}+C_{11}) + A_{12}(B_{12}+C_{12}) + A_{13}(B_{13}+C_{13}) \\
&+ A_{21}(B_{21}+C_{21}) + ... \\
&= \\
&+ A_{11}B_{11} + A_{11}C_{11} + A_{12}B_{12} + A_{12}C_{12} + A_{13}B_{13} + A_{13}C_{13} \\
&+ A_{21}B_{21} + A_{21}C_{21} + ... \\
\\
A:(B+C) &= A:B + A:C \\
&+ A_{11}B_{11} + A_{11}C_{11} + A_{12}B_{12} + A_{12}C_{12} + A_{13}B_{13} + A_{13}C_{13} = \\
&+ A_{21}B_{21} + A_{21}C_{21} + ... \\
\\
A:(B+C) &= A:B + A:C
\end{aligned}
\tag{37}
$$

References

[1] A. Hellsten. Some improvements in menter's k-omega-sst turbulence model. *AIAA-98-2554*, 1998.

[2] F. R. Menter, M. Kuntz, and R. Langtry. Ten Years of Industrial Experience with the SST Turbulence Model. *Turbulence, Heat and Mass Transfer 4*, pages 625 – 632, 2003.

[3] NASA. Menter shear stress transport model, 2016. [Online; accessed 14-January-2016].