7.2.3 Model coefficients

The coefficients for the RAS turbulence models are given default values in their respective source code. If the user wishes to override these default values, then they can do so by adding a sub-dictionary entry to the `RAS` sub-dictionary file, whose keyword name is that of the model with `Coeffs` appended, *e.g.* `kEpsilonCoeffs` for the `kEpsilon` model. If the `printCoeffs` switch is `on` in the `RAS` sub-dictionary, an example of the relevant `...Coeffs` dictionary is printed to standard output when the model is created at the beginning of a run. The user can simply copy this into the `RAS` sub-dictionary file and edit the entries as required.

7.2.4 Wall functions

A range of wall function models is available in OpenFOAM that are applied as boundary conditions on individual patches. This enables different wall function models to be applied to different wall regions. The choice of wall function model is specified through the turbulent viscosity field ν_t in the *0/nut* file. For example, a *0/nut* file:

```
17
18  dimensions      [0 2 -1 0 0 0 0];
19
20  internalField   uniform 0;
21
22  boundaryField
23  {
24      movingWall
25      {
26          type            nutkWallFunction;
27          value           uniform 0;
28      }
29      fixedWalls
30      {
31          type            nutkWallFunction;
32          value           uniform 0;
33      }
34      frontAndBack
35      {
36          type            empty;
37      }
38  }
39
40
41  // ************************************************************************* //
```

There are a number of wall function models available in the release, *e.g.* `nutWallFunction`, `nutRoughWallFunction`, `nutUSpaldingWallFunction`, `nutkWallFunction` and `nutkAtmWallFunction`. The user can get the full list of wall function models using foamInfo:

```
foamInfo wallFunction
```

Within each wall function boundary condition the user can over-ride default settings for E, κ and C_μ through optional `E`, `kappa` and `Cmu` keyword entries.

Having selected the particular wall functions on various patches in the *nut*/*mut* file, the user should select epsilonWallFunction on corresponding patches in the *epsilon* field and kqRwallFunction on corresponding patches in the turbulent fields *k*, *q* and *R*.

7.3 Transport/rheology models

In OpenFOAM, solvers that do not include energy/heat, include a library of models for viscosity ν. The models typically relate viscosity to strain rate $\dot{\gamma}$ and are specified by the

user in the *transportProperties* dictionary. The available models are listed in the following sections.

7.3.1 Newtonian model

The Newtonian model assumes ν is constant. Viscosity is specified by a dimensionedScalar `nu` in *transportProperties*, *e.g.*

```
transportModel Newtonian;

nu              [ 0 2 -1 0 0 0 0 ] 1.5e-05;
```

Note the units for kinematic viscosity are L^2/T.

7.3.2 Bird-Carreau model

The Bird-Carreau model is:

$$\nu = \nu_\infty + (\nu_0 - \nu_\infty)\left[1 + (k\dot{\gamma})^a\right]^{(n-1)/a} \tag{7.20}$$

where the coefficient a has a default value of 2. An example specification of the model in *transportProperties* is:

```
transportModel BirdCarreau;
BirdCarreauCoeffs
{
    nu0       [ 0 2 -1 0 0 0 0 ] 1e-03;
    nuInf     [ 0 2 -1 0 0 0 0 ] 1e-05;
    k         [ 0 0  1 0 0 0 0 ] 1;
    n         [ 0 0  0 0 0 0 0 ] 0.5;
}
```

7.3.3 Cross Power Law model

The Cross Power Law model is:

$$\nu = \nu_\infty + \frac{\nu_0 - \nu_\infty}{1 + (m\dot{\gamma})^n} \tag{7.21}$$

An example specification of the model in *transportProperties* is:

```
transportModel CrossPowerLaw;
CrossPowerLawCoeffs
{
    nu0       [ 0 2 -1 0 0 0 0 ] 1e-03;
    nuInf     [ 0 2 -1 0 0 0 0 ] 1e-05;
    m         [ 0 0  1 0 0 0 0 ] 1;
    n         [ 0 0  0 0 0 0 0 ] 0.5;
}
```

7.3.4 Power Law model

The Power Law model provides a function for viscosity, limited by minimum and maximum values, $\nu_{\min}$ and $\nu_{\max}$ respectively. The function is:

$$\nu = k\dot{\gamma}^{n-1} \quad \nu_{\min} \leq \nu \leq \nu_{\max} \tag{7.22}$$

An example specification of the model in *transportProperties* is:

```
transportModel powerLaw;
powerLawCoeffs
{
    nuMax     [ 0 2 -1 0 0 0 0 ] 1e-03;
    nuMin     [ 0 2 -1 0 0 0 0 ] 1e-05;
    k         [ 0 2 -1 0 0 0 0 ] 1e-05;
    n         [ 0 0  0 0 0 0 0 ] 1;
}
```

7.3.5 Herschel-Bulkley model

The Herschel-Bulkley model combines the effects of Bingham plastic and power-law behavior in a fluid. For low strain rates, the material is modelled as a very viscous fluid with viscosity ν_0. Beyond a threshold in strain-rate corresponding to threshold stress τ_0, the viscosity is described by a power law. The model is:

$$\nu = \min\left(\nu_0, \tau_0/\dot{\gamma} + k\dot{\gamma}^{n-1}\right) \tag{7.23}$$

An example specification of the model in *transportProperties* is:

```
transportModel HerschelBulkley;
HerschelBulkleyCoeffs
{
    nu0       [ 0 2 -1 0 0 0 0 ] 1e-03;
    tau0      [ 0 2 -2 0 0 0 0 ] 1;
    k         [ 0 2 -1 0 0 0 0 ] 1e-05;
    n         [ 0 0  0 0 0 0 0 ] 1;
}
```

7.3.6 Casson model

The Casson model is a basic model used in blood rheology that specifies minimum and maximum viscosities, $\nu_{\min}$ and $\nu_{\max}$ respectively. Beyond a threshold in strain-rate corresponding to threshold stress τ_0, the viscosity is described by a "square-root" relationship. The model is:

$$\nu = \left(\sqrt{\tau_0/\dot{\gamma}} + \sqrt{m}\right)^2 \quad \nu_{\min} \leq \nu \leq \nu_{\max} \tag{7.24}$$

An example specification of model parameters for blood is:

```
transportModel Casson;
CassonCoeffs
{
    m        [ 0 2 -1 0 0 0 0 ] 3.934986e-6;
    tau0     [ 0 2 -2 0 0 0 0 ] 2.9032e-6;
    nuMax    [ 0 2 -1 0 0 0 0 ] 13.3333e-6;
    nuMin    [ 0 2 -1 0 0 0 0 ] 3.9047e-6;
}
```

7.3.7 General strain-rate function

A `strainRateFunction` model exists that allows a user to specify viscosity as a function of strain rate at run-time. It uses the same Function1 functionality to specify the function of strain-rate, used by time varying properties in boundary conditions described in section 5.2.3.4. An example specification of the model in *transportProperties* is shown below using the `polynomial` function:

```
transportModel  strainRateFunction;
strainRateFunctionCoeffs
{
    function polynomial ((0 0.1) (1 1.3));
}
```