viscousHeatingSolver

Adding temperature transport and viscous heating to simpleFoam

Solver and Test Case

Martin Becker

martin_becker@dhcae-tools.de

11/09/2012

Abstract

The modifications of the simpleFoam solver to implement temperature transport and viscous heating are described. The boundary conditions and settings in fvSchemes and fvSolutions are described. The analytical results are compared to the simulation results.

The tutorial case and the modifications are useful for the simulation of high viscous polymer flow.

Trademarks

OPENFOAM and OpenCFD are registered trademarks of ESI Group. This offering is not approved or endorsed by ESI Group, the producer of the OpenFOAM® software and owner of the OPENFOAM® and OpenCFD® trade marks.

1 Introduction

simpleFoam is the OpenFOAM's standard solver for incompressible, isothermal and steady state flow. You can use Newtonian fluid behaviour or non-Newtonian material models like BirdCarreau or CrossPowerLaw. Optionally there are different turbulence models available.
With these basic characteristics simpleFoam qualifies itself to be the foundation for simulating high viscous polymers. An excellent application area is for example polymer extrusion, where non-Newtonian materials are formed in a steady state production process.
An interesting aspect in the extrusion process is the transport of temperature, and for high viscous polymers especially the viscous heating or viscous dissipation. Both features are added to simpleFoam and they are discussed in this paper.

2 Modifications of the simpleFoam solver

2.1 Compilation and usage

The new solver is name viscousHeatingSolver. It is directly derived from the original simpleFoam solver of OpenFOAM-2.1.x. To compile the solver you therefor need a working installation of OpenFOAM-2.1.x.
After unpacking the solver to your user OpenFOAM user directory you can compile it with "wmake". The solver will be placed in your $(FOAM_USER_APPBIN) directory.
The attached case "case_capillary_viscousHeating" contains an "Allrun.sh" script. You can run the script to see the solver in action.

2.2 Modifications of createFields.H

For the temperature transport the thermal diffusivity coefficient DT is used, and the viscous dissipation needs the specific heat capacity c. Both are assumed to be constant and they are read within the "createFields.H" file:

```
// thermal diffusivity [DT]
dimensionedScalar DT_(laminarTransport.lookup("DT"));
// specific heat capacity [c]
dimensionedScalar c_(laminarTransport.lookup("c"));
```

Now these constants are globally available in the solver and might be used for other purposes.

2.3 The new TEqn.H file

The solution of the temperature transport equation is done in a new file named "TEqn.H".
The computation of the viscous heating term has been unrolled: gradU and tau are calculated explicitly before they are used in the TEqn itself. Be aware of the brackets around the lines of code: they reduce the memory usage because the additional volScalarField and volTensorFields are deleted automatically when the solver leaves this section.
The diffusivity coefficient DT is used in the laplacian term of the temperature equation.

```
{
    volTensorField gradU = fvc::grad(U);
    tmp <volScalarField> nu = laminarTransport.nu();
    tmp <volTensorField> tau = nu * (gradU + gradU.T());
    fvScalarMatrix TEqn
    (
        fvm::div(phi, T)
      - fvm::laplacian(DT_, T)
      == (1/c_)*(tau && gradU) // viscous heat dissipation term
    );
    TEqn.relax();
    TEqn.solve().initialResidual();
}
```

2.4 Modifications of mySimpleFoam.C

The new TEqn.H file must be included into the main solver. An appropriate location is after the U update and before the pEqn:

```
// --- Pressure-velocity SIMPLE corrector
{
    #include "UEqn.H"
    #include "TEqn.H"
    #include "pEqn.H"
}
```

3 Case setup

3.1 constant/transportProperties

As an example we run a simulation on a simplified capillary rheometer. We use a fictious Newtonian fluid here with these material properties:

Physical parameter	Symbol	Value
density	ρ	$1380\frac{kg}{m^3}$
thermal conductivity	κ	$0.166\frac{W}{K \cdot m}$
specific heat capacity	c	$1530\frac{J}{kg \cdot K}$
dynamic viscosity	μ	$1500Pa \cdot s$

The thermal diffusivity coefficient DT results from:

$$DT = \frac{\kappa}{\rho \cdot c} = \frac{0.166\frac{W}{K \cdot m}}{1380\frac{kg}{m^3} \cdot 1530\frac{J}{kg \cdot K}} = 7.8621e^{-8}\frac{m^2}{s}$$

An the kinematic viscosity used by the incompressible OpenFOAM solvers is calculated this way:

$$\nu = \frac{\mu}{\rho} = \frac{1500Pa \cdot s}{1380\frac{kg}{m^3}} = 1.087\frac{m^2}{s}$$

This results in the following transportProperties file:

```
transportModel  Newtonian;
nu   nu   [0 2 -1 0 0 0 0]  1.087;
DT   DT   [0 2 -1 0 0 0 0]  7.8621e-8;
rho  rho  [1 -3 0 0 0 0 0]  1380.0;
c    c    [0 2 -2 -1 0 0 0] 1530.0;
```

3.2 Boundary conditions

File	Patch	Boundary condition	Comment
0/T	inlet	fixedValue, 503.15K	
	wall, outlet	zeroGradient	
0/U	inlet	fixedValue, (0 0 0.0006666)	results in $4\frac{m}{min}$block velocity @ outlet
	wall	fixedValue, (0 0 0)	
	outlet	zeroGradient	
0/p	outlet	fixedValue, 0	
	inlet, wall	zeroGradient	

3.3 fvSchemes and fvSolution

The new equation system for T needs some new entries in the fvSchemes file:

Scheme	Entry
gradSchemes	grad(T) Gauss linear;
divSchemes	div(phi,T) Gauss linearUpwind grad(T);
laplacianSchemes	laplacian(DT,T) Gauss linear corrected;

In the fvSolution file a linear equation system solver must be set:

```
T
{
    solver          smoothSolver;
    smoother        DILUGaussSeidel;
    tolerance       1e-06;
    relTol          0.01;
    maxIter         100;
}
```

A relaxation factor should be added to the relaxationFactors dictionary:

```
relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        U               0.7;
        T               0.9;
    }
}
```

On optional entry to the SIMPLE/residualControl looks like this:

```
residualControl
{
    U               1e-5;
    p               1e-5;
    T               1e-5;
}
```

4 Analytical vs. simulation results

The viscous heating can be calculated by the knowledge of the pressure drop:

$$\Delta T = \frac{p_{outlet} - p_{inlet}}{\rho \cdot c} \tag{1}$$

After running the simulation you can calculate the kinematic pressure with the patchAverage utility:

```
patchAverage p inlet
```

The kinematic pressure drop is $11891.7 \frac{m^2}{s^2}$ in the attached case.
The flow rate averaged temperature at the outlet is written out with a simpleFunctionsObject in the controlDict file:

```
functions
{
    massFlowAverageT
    {
        type patchMassFlowAverage;
        functionObjectLibs ( "libsimpleFunctionObjects.so" );
        fields ( T );
        patches
        ( outlet );
        factor 1.0;
        verbose true;
    }
}
```

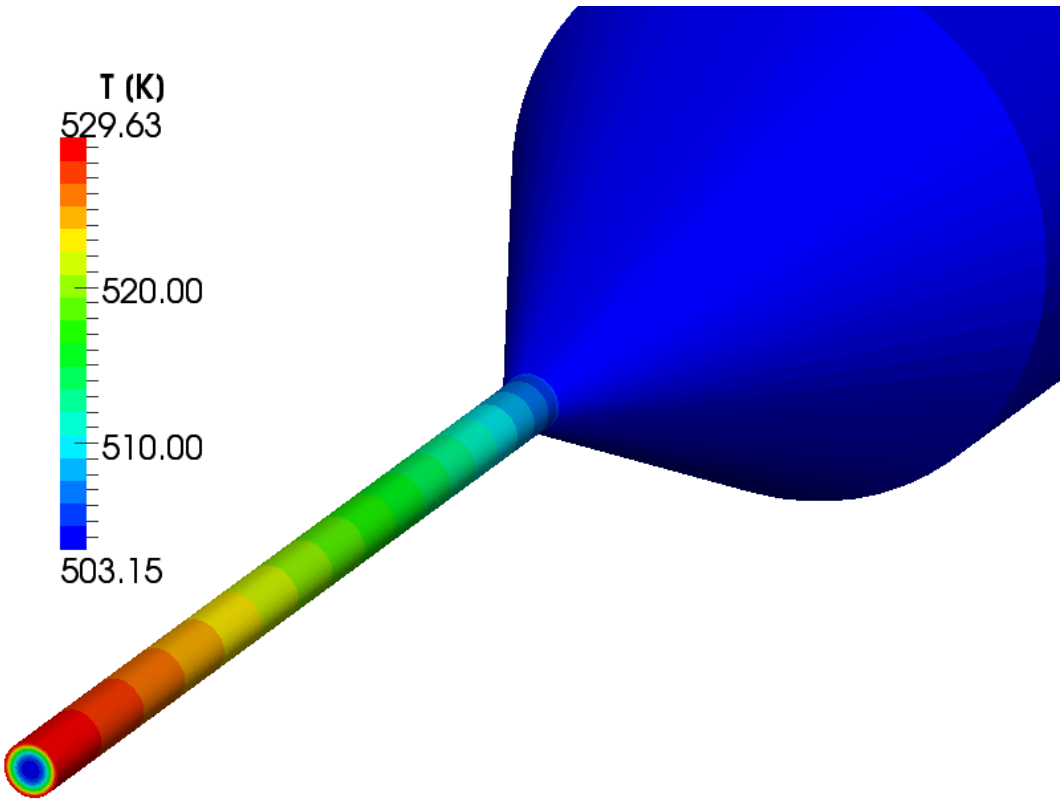

Figure 1: Temperature [K]

This function gives a temperature at the outlet of $510.896K$, id est a $\Delta T = T_{outlet} - T_{inlet} = 510.896K - 503.15K = 7.746K$. The analytical solution from the formula 1 gives $\Delta T = \frac{11891.7 \frac{m^2}{s^2}}{1530 \frac{J}{kg \cdot K}} = 7.772K$.

5 Where to go from here?

There are numerous interesting aspects in polymer processing simulations that can be added to simpleFoam or other OpenFOAM solver, for example:

- temperature dependend, time dependend or pressure dependend viscosity models to improve the simulation results
- residence time analysis to identify thermal stress on the polymer
- flow channel optimization strategies
- shear rates and shear stresses analysis
- and much more...

DHCAE Tools UG (limited liability company)
Friedrich-Ebert-Str. 368
47800 Krefeld
Germany
Phone +49 2151 821493
Fax +49 2151 821494