OpenFOAM Tutorials

2 Add temperature to solvers

2.1 Add the heat transport equation to the `icoFoam` solver, in order to calculate how the flow can affect the heat transfer between two heated walls. The heat transport equation is

$$\frac{\partial T}{\partial t} + (\boldsymbol{\nabla} \cdot \boldsymbol{V})T = \alpha \nabla^2 T$$

where α is the themal diffusivity.

Copy the `icoFoam` solver to your OpenFOAM project directory, inside a folder named `applications`, where you can store all your modified OpenFOAM solvers. If you haven't already created such directory, either do it manually, or follow the commands

```
$ mkdir -p $WM_PROJECT_USER_DIR/applications
$ cp -r $FOAM_SOLVERS/incompressible/icoFoam $WM_PROJECT_USER_DIR/applications/myIcoFoam
$ cd $WM_PROJECT_USER_DIR/applications/solvers/myIcoFoam
```

Rename the `icoFoam.C` file to `myIcoFoam.C`. Now go into the Make subdirectory and open the 'files' file with a text editor. Change it to read:

```
myIcoFoam.C

EXE = $(FOAM_USER_APPBIN)/myIcoFoam
```

You can delete old binaries, if they exist, using the command

```
$ wclean
```

To compile the new solver, run the command

```
$ wmake
```

If everything worked correctly, your new solver binary should be present in the `FOAM_USER_APPBIN` directory. Check this with:

```
$ ls $FOAM_USER_APPBIN
```

This will show the steps to add another field variable to a solver. Open the `myIcoFoam.C` with a text editor. Following the flow of the `myIcoFoam` program, one notices that the header file `createFields.H` is called prior to the solution loop. This file was copied with the solver and has the specific information pertaining to what variables will be solved. Inside the `createFields.H` file, the first items loaded is the kinematic viscosity from the `transportProperties` dictionary file. We will add a new transport property related to the thermal diffusivity which will be denoted as `alpha`. Make the following edits:

```
Info<< "Reading transportProperties\n" << endl;

IOdictionary transportProperties
(
   IOobject
   (
      "transportProperties",
      runTime.constant(),
      mesh,
      IOobject::MUST_READ,
      IOobject::NO_WRITE
   )
);

dimensionedScalar nu
(
   transportProperties.lookup("nu")
);
```

```
//Add here...
dimensionedScalar alpha
(
   transportProperties.lookup("alpha")
);
//Done for now...
```

Later on, we will need to edit the `transportProperties` dictionary file to reflect this change. Moreover, in the same file, add a `volScalarField`, similar to the pressure field, that represents the temperature.

```
Info<< "Reading field T\n" <<endl;

volScalarField T
(
   IOobject
   (
      "T",
      runTime.timeName(),
      mesh,
      IOobject::MUST_READ,
      IOobject::AUTO_WRITE
   ),
   mesh
);
```

Save these changes. You've completed adding a new scalar field variable 'T' and a new constant 'alpha'.

The next step is to add a new equation describing the transport of the temperature. Return to editing the `myIcoFoam.C` file. Because the temperature transport depends on the velocity field, we will add the equation after the momentum equation is solved (after the PISO loop), but before the time step is written. Edit your file so it looks like this:

```
         #include "continuityErrs.H"

         U -= rUA*fvc::grad(p);
         U.correctBoundaryConditions();
      }

      //add these lines...
      fvScalarMatrix TEqn
      (
         fvm::ddt(T)
         + fvm::div(phi, T)
         - fvm::laplacian(alpha, T)
      );

      TEqn.solve();
      //done adding lines...

      runTime.write();
```

These lines add a new equation for the temperature and make use of the face flux variable, phi, which is already used in the momentum equation solution.

Save your changes and run wmake to compile the solver:

```
$ wmake
```

2.2 The next step in this process is to test the new solver. This will first be accomplished by modifying the existing cavity tutorial. Copy a cavity case from the tutorials folder into your OpenFOAM directory. Open the `transportProperties` dictionary in your favorite editor and add the following line under the definition of nu:

```
alpha         alpha [0 2 -1 0 0 0 0] 0.002;
```

In the `./0/` folder, create a file for the initial and boundary conditions of temperature:

```
FoamFile
{
   version       2.0;
   format        ascii;
   class         volScalarField;
   object        T;
}
//*********************************//

dimensions            [0 0 0 1 0 0 0];

internalField         uniform 300;

boundaryField
{
   movingWall
   {
      type            fixedValue;
      value           uniform 350;
   }

   fixedWalls
   {
      type            fixedValue;
      value           uniform 300;
   }

   frontAndBack
   {
      type            empty;
   }
}
```

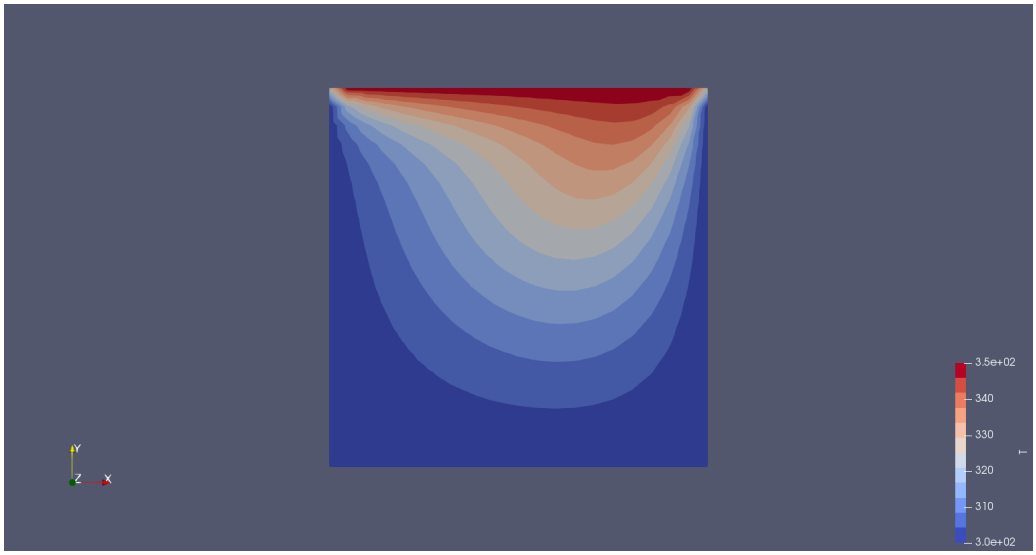
The next thing tha must be done is to choose discretization scemes and solution algorithms for the extra energy equation. By adding a new equation to solve, we need to tell OpenFOAM what discretization schemes to apply to the equations. This is done in the `fvSchemes` dictionary. Open the 'fvSchemes' dictionary. Now, there are two main items added in the thermal transport equation above: a divergence term and a laplacian. Under each of these sections in the dictionary, we need to add terms which match what was added in the solver source code. Edit your file so it includes the following:

```
divSchemes
{
   default                none;
   div(phi,U)             Gauss linear;
   div(phi,T)             Gauss upwind;
}

laplacianSchemes
{
   default                none;
   laplacian(nu,U)        Gauss linear corrected;
   laplacian((1|A(U)),p)  Gauss linear corrected;
   laplacian(alpha,T)     Gauss linear corrected;
}
```

Alternatively, a scheme can be added to the 'default' field instead of adding specific ones below. Next, open the `fvSolution` dictionary. It should look like this:

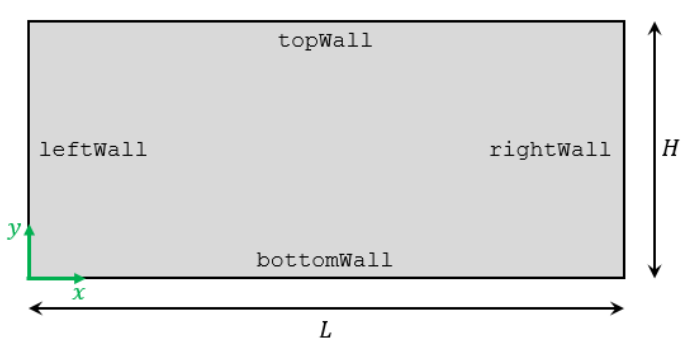
```
solvers
{
   p
   {
      solver            PCG;
      preconditioner    DIC;
      tolerance         1e-06;
      relTol            0.05;
   }

   pFinal
   {
      $p;
      relTol            0;
   }

   U
   {
      solver            smoothSolver;
      smoother          symGaussSeidel;
      tolerance         1e-05;
      relTol            0;
   }
   //add this...
   T
   {
      solver            BICCG;
      preconditioner    DILU;
      tolerance         1e-7;
      relTol            0;
   };
   //done editing...
}
```

After successfully running the case, the temperature field should look similar to the following picture



2.3 Simulate the thermal laminar 3D pipe flow of the previous tutorial, by adding a temperature field, similarly with the previous step. Define two opposite walls with temperatures 300 K and 350 K respectively, while the other two will be fully insulated.

2.4 Create a new solver, called `laplaceFoam` that solves the Laplace equation in a rectangular flat plate, with boundary and initial conditions as shown in the next figure. Use the quantities $T_w = 100\ K$, $L = 2\ m$ and $H = 1\ m$.



Laplace Equation: $\nabla^2 T = 0$

Boundary Conditions

$T(0, y, t) = T_w$

$T(L, y, t) = 0$

$T(x, 0, t) = 0$

$T(x, H, t) = 0$

Initial Condition

$T(x, y, 0) = 0$

For the creation of the new solver, copy the existing solver `electrostaticFoam` into the `applications` directory, in your `Home` folder, and change the name in all the appropriate files, as described in Step 2.

Modify the `laplaceFoam.C` and the `createFields.H` files, in order to only solve the Laplace equation for the `volScalarField` T. Delete any unnecessary commands, equations and variables inside the solver and compile the solver.

Create the case and test the validity of your results.