Implement a passive scalar transport solver

Implement a passive scalar transport solver

Contents

- You will add an additional scalar transport equation to an existing solver.

Prerequisites

- You are familiar with the directory structure of OpenFOAM applications.
- You are familiar with user compilation procedures of applications.
- You are familiar with the fundamental high-level components of application codes, and how new classes can be introduced to an application.

Learning outcomes

- You will practice high-level coding and modification of solvers.
- You will adapt case set-ups according to the new solver.
- You will improve your understanding of classes and object orientation, from a high-level perspective.

Implement a passive scalar transport solver

Very important note:

- The instructions in these slides are for `icoFoam` and the `icoFoam/cavity` case, but:
- To make it slightly more interesting you should use the instructions to instead implement this for `pimpleFoam` and the `pimpleFoam/RAS/pitzDaily` case.
- Assume that you want to add a passive scalar field that you can use to visualize how far into the computational domain the flow has passed for each time step. Thus, initialize the internal field of the scalar to zero and set the inlet value to one. When the value of the scalar is one in the entire domain, you have an indication that the flow has passed through the entire domain.
- Does this mean that you have reached a point where the flow is fully developed?

Note that you will be asked to pack up your final cleaned-up directories and submit them for assessment of completion:
`passiveScalarPimpleFoam`
`passiveScalarPitzDaily`
(not the `icoFoam/cavity` alternative)

Copy the icoFoam solver, rename it, and test that it still compiles

- We copy the `icoFoam` solver and put it in our `$WM_PROJECT_USER_DIR` with the same file structure as in the OpenFOAM installation:

```
foam
cp -r --parents applications/solvers/incompressible/icoFoam $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/incompressible
mv icoFoam passiveScalarIcoFoam
cd passiveScalarIcoFoam
wclean
mv icoFoam.C passiveScalarIcoFoam.C
```

- Modify `Make/files` (most portable way):

```
string="passiveScalarIcoFoam.C\nEXE = \$(FOAM_USER_APPBIN)/passiveScalarIcoFoam"
printf "%b\n" "$string" > Make/files
```

  Make sure that you understand what this command does, and why it is done!

- Compile with `wmake` in the `passiveScalarIcoFoam` directory. `rehash` if necessary.

Test on the cavity case

We will quickly visit the run directory to test...

```
pushd $FOAM_RUN #so that we can easily go back to the current directory
rm -r cavity
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity .
blockMesh -case cavity
passiveScalarIcoFoam -case cavity
```

After checking that it worked, go back to the `passiveScalarIcoFoam` directory:

```
popd #brings you back to the directory where you typed the pushd command
```

You can also do everything 'remotely':

```
rm -r $FOAM_RUN/cavity
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity $FOAM_RUN
blockMesh -case $FOAM_RUN/cavity
passiveScalarIcoFoam -case $FOAM_RUN/cavity
```

Add a passive scalar transport equation

- Let's add, to `passiveScalarIcoFoam`, the passive scalar transport equation

$$\frac{\partial s}{\partial t} + \nabla \cdot (\mathbf{u}\; s) = 0$$

- Modify the solver according to:
  - Create `volumeScalarField s` (do the same as for `p` in `createFields.H`, since both are scalar fields)
  - Add the equation `solve(fvm::ddt(s) + fvm::div(phi, s));` before `runTime.write();` in `passiveScalarIcoFoam.C`.
- Compile `passiveScalarIcoFoam` using `wmake`

Make sure that you understand why those modifications are made, and why the pieces of code are put at those exact locations! Why don't we have to do more modifications?

Modify the icoFoam/cavity case

- Set up the case according to:
  - Use the `icoFoam/cavity` case as a base:
    ```
    run
    cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity passiveScalarCavity
    cd passiveScalarCavity
    ```
  - Copy the `0/p` file to `0/s` and modify `p` to `s` in that file. Choose approprate dimensions for the scalar field (not important now).
  - In `fvSchemes`, add (if you don't, it will complain):
    ```
        div(phi,s)        Gauss linearUpwind grad(U);
    ```
  - In `fvSolution`, copy the solution settings from `U` (since the equations for velocity and `s` are similar), and just change `U` to `s`. (if you use `PCG`, as for `p`, it will complain - try it yourself!)

Make sure that you understand why those modifications are made! Why don't we have to do more modifications?

Initialize, run and post-process the case

- Initialize `s`:
  - `cp $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak/damBreak/system/setFieldsDict system`
  - Set `defaultFieldValues`:
    `volScalarFieldValue s 0`
  - Modify the bounding box to:
    `box (0.03 0.03 -1) (0.06 0.06 1);`
  - Set `fieldValues`:
    `volScalarFieldValue s 1`
- Run the case:

```
blockMesh
setFields
passiveScalarIcoFoam >& log
```

  `paraFoam` - mark `s` in Volume Fields, color by `s` (cell value) and run an animation.
- You can see that although there is no diffusion term in the equation, there is massive diffusion in the results. This is due to mesh resolution, numerical scheme etc. The `interfoam` solver has a special treatment to reduce this kind of diffusion.