pyFoam, a user contribution

- Described in the OpenFOAM Wiki:
  `http://openfoamwiki.net/index.php/Contrib_PyFoam`
  (`openfoamwiki.net`, click on `Contributions`, `Other`,
  `Libraries for other languages`, `PyFoam`)
- Available through SVN at SourceForge
- `pyFoam` is NOT in the OpenFOAM distribution! You will have to install it separately!
- Features (some examples):
  - Uses OpenFOAM libraries to connect to OpenFOAM.
  - Execute applications, and analyse and modify their output.
  - Run lots of parameter variations of the same case.
  - Manipulate OpenFOAM dictionaries, such as for setting up new cases.
  - Plot residuals on the fly using Gnuplot.
  - View the block structure of a `blockMeshDict` (requires VTK, doesn't work here).
- We will now use `pyFoam` to set up dictionaries and to plot information in the `log` file.

Setting up a case from scratch

Basic method:

- Find out which solver you need to use for the specific problem.
- Copy a tutorial for that specific solver to your run directory:

```
cp -r $FOAM_TUTORIALS/simpleFoam/pitzDaily $FOAM_RUN/simpleElbow
cd $FOAM_RUN/simpleElbow
```

- Modify all the dictionaries according to how you want to run your case (keep default now).
- Modify the mesh by editing `blockMeshDict` and running `blockMesh`, or use a third-party mesh-generator and a converter utility (`fluentMeshToFoam` usually works). Here:
  `cp $FOAM_TUTORIALS/icoFoam/elbow/elbow.msh .`
  `fluentMeshToFoam elbow.msh`
- Now, the patch names in the time directory is probably not in accordance with the ones in `constant/polyMesh/boundary`
- Edit all the dictionaries in the time directory so that all the patch names in `constant/polyMesh/boundary` are present. Also set the appropriate boundary condition for each patch. This is a lot of work! There is however an option...

pyFoamCreateBoundaryPatches.py

- We will use the `pyFoamCreateBoundaryPatches.py` to set up the time dictionaries.
- For help:
  `pyFoamCreateBoundaryPatches.py --help`
- Modify the `0/U,p,k,epsilon` and `controlDict` dictionaries (get at course homepage):

```
pyFoamCreateBoundaryPatches.py --verbose --clear-unused 0/U
pyFoamCreateBoundaryPatches.py --verbose --overwrite --filter="wall.+" \
      --default="{'type':'fixedValue','value':'uniform (0 0 0)'}" 0/U
pyFoamCreateBoundaryPatches.py --verbose --overwrite --filter="velocity-inlet-5" \
      --default="{'type':'fixedValue','value':'uniform (1 0 0)'}" 0/U
pyFoamCreateBoundaryPatches.py --verbose --overwrite --filter="velocity-inlet-6" \
      --default="{'type':'fixedValue','value':'uniform (0 3 0)'}" 0/U
pyFoamCreateBoundaryPatches.py --verbose --clear-unused 0/p
pyFoamCreateBoundaryPatches.py --verbose --overwrite --clear-unused --filter="pressure.+" \
      --default="{'type':'fixedValue','value':'uniform 0'}" 0/p
pyFoamCreateBoundaryPatches.py --verbose --clear-unused 0/k
pyFoamCreateBoundaryPatches.py --verbose --overwrite --filter="velocity-inlet.+" \
      --default="{'type':'turbulentIntensityKineticEnergyInlet','intensity':'0.1', \
      'value':'uniform 0.375'}" 0/k
pyFoamCreateBoundaryPatches.py --verbose --clear-unused 0/epsilon
pyFoamCreateBoundaryPatches.py --verbose --overwrite --filter="velocity-inlet.+" \
      --default="{'type':'turbulentMixingLengthDissipationRateInlet', \
      'mixingLength': '0.05', 'value':'uniform 14.855'}" 0/epsilon
pyFoamWriteDictionary.py system/controlDict endTime 50
```

- This also seems quite complicated, but if you use consistent naming, this can be re-used for other cases.

Clean up, run and plot residuals on the fly

- We did not update `0/nuTilda` and `0/R` since we will use the `kEpsilon` model, which does not use them. Delete them:
  `rm 0/nuTilda`
  `rm 0/R`
- Run `simpleFoam` on the case and plot residuals on-the-fly:
  - Open a separate terminal window for the plotter and type:
    `cd $FOAM_RUN/simpleElbow`
    `rm log` (make sure to start from scratch)
    `touch log` (this generates a new empty `log` file)
    `pyFoamPlotWatcher.py log` (The plotter will now point at the beginning of the log file and wait for a solver to add information to the `log` file)
  - Run the `simpleFoam` solver in the first terminal window:
    `simpleFoam >&log`
- You will now have plots of residuals, continuity error and bounding for the initial 50 iterations. In the terminal window where you ran the plotter you will have the entire log file printed.

Continue plotting and close down the plotter

- Modify `endTime` to 1000 in `system/controlDict`, run `simpleFoam >& log`, and the plotting will continue. Note that the plotter will remember the line-number from the previous run as long as you do not restart the plotter with an *empty* `log` file. In this case, `simpleFoam` will start writing in the beginning of the `log` file, and the plotter will continue plotting when we have reached beyond the 50th iteration.

- Kill the plotter by doing `CTRL-c` in the plotter terminal window.
  If it doesn't work you will have to kill the process. Find the PID (Process ID) from the terminal window where you started `simpleFoam` by typing:
  `ps -ef | grep pyFoamPlotWatcher.py`
  This will give something similar to:

```
hani 22729 25593 0 11:06 pts/235 00:00:00 /usr/bin/python \
     /chalmers/sw/unsup/OpenFOAM/ThirdParty/PyFoam/bin/pyFoamPlotWatcher.py log
hani 31269 12789 0 11:15 pts/254 00:00:00 grep pyFoamPlotWatcher.py
```

  The PID of the plotter is the first number of the line with `pyFoamPlotWatcher.py log`, in this case: `22729`. Kill it by typing:
  `kill 22729`

pyFoamPlotRunner.py

- `pyFoamPlotRunner.py` does basically the same thing as `pyFoamPlotWatcher.py`, but it also starts the solver:
  `pyFoamPlotRunner.py simpleFoam`
- `pyFoamPlotRunner.py --help` for more information
- One difference is that you are not able to run the plotting on another computer, so the processes will compete for the same CPU power.
- It is further not possible to turn on and off the plotter itself.

Find other installed pyFoam scripts

- Find out which file you are actually running by typing:
  `which pyFoamPlotWatcher.py`
  This should make you aware of a directory named:
  `$WM_THIRD_PARTY_DIR/PyFoam/bin`
  where you can find all the `pyFoam` scripts in the `pyFoam` distribution. NOTE that this location was decided by myself, as I installed `pyFoam` separate from the OpenFOAM installation!
- Use the `--help` flag to get more information on each script.
- Read more at: `http://openfoamwiki.net/index.php/Contrib_PyFoam`
- `pyFoam` is OpenSource, so you can modify it according to your needs.
- At the same time as you dig into `pyFoam`, you will also learn how to do Python script programming.

An alternative for modifying dictionaries

- As always, you can do things in different ways...
- See the `chtMultiRegionFoam/multiRegionHeater` for a `changeDictionaryDict`, which can be used together with `changeDictionary` to set up dictionaries.
- Search the Forum for `changeDictionary`