swak4Foam and PyFoam

One hot case

Bernhard F.W. Gschaider

HFD Research GesmbH

Duisburg, Germany
23. July 2019

Outline I

Outline II

- Evaluations on boundaries

5 Adding more features
- Smoothing the floor temperature
- Backport of `lumped` condition
- Variable heat transfer

6 Conclusions

Outline

Outline

The topic

- Two programs/libraries/toolkits
  - swak4Foam
  - PyFoam
- Very different
  - What they have in common
    - Used with OpenFOAM
    - Written by me
- I usually use them together
  - Because in their difference they complement each other
- Therefor this presentation tries to introduce them together

Intended audience and aim

- Intended audience for this presentation:
  - people who already worked a bit with OpenFOAM

    worked a bit means: been through the tutorials and set up a case on their own
  - have heard that `PyFoam` and `swak4Foam` exist
- Aim of the presentation
  - Enable user to start using PyFoam and swak4Foam
  - No programming
- The presentation is designed so that all steps can be reproduced using the information on the slides
  - No training files are provided

Format of the presentation

- This is a hands-on tutorial
- We will use a standard tutorial case
- Modify it till it doesn't look like the original
- No additional files are needed
  - Everything you have to enter will be spelled out on the slides
  - But to be sure: intermediate states will be available as download

Limitation

- In 1 hour we can only give superficial overview of the two packages
    - It is not sure whether we'll even be able to complete it
        - I will "speed" through things that are not as interesting
        - If you've got questions about the "gaps": I'll be here all week
- For a complete reference of the swak-expressions have a look at the *Incomplete reference guide* that comes with `swak`
    - Expressions are completely described
    - Almost everything else is missing

Outline

Bernhard Gschaider

- Working with OpenFOAM™ since it was released
  - Still have to look up things in `Doxygen`
- I am not a core developer
  - But I don't consider myself to be an *Enthusiast*
- My involvement in the OpenFOAM™-community
  - Janitor of the `openfoamwiki.net`
  - Author of two additions for OpenFOAM™
    - swak4foam Toolbox to avoid the need for C++-programming
    - PyFoam Python-library to manipulate OpenFOAM™ cases and assist in executing them
  - In the admin-team of `foam-extend`
  - Organizing committee for the OpenFOAM™ *Workshop*
- The community-activies are not my main work but *collateral damage* from my real work at ...

Heinemann Fluid Dynamics Research GmbH

The company

- Subsidary company of *Heinemann Oil*
  - Reservoir Engineering
  - Reservoir management

Description

- Located in Leoben and Vienna, Austria
- Works on
  - Fluid simulations
    - OpenFOAM™ and Closed Source
  - Software development for CFD
    - mainly OpenFOAM™
- Industries we worked for
  - Automotive
  - Processing
  - ...

Outline

What is PyFoam

- PyFoam is a library for
  - Manipulating OpenFOAM-cases
  - Controlling OpenFOAM-runs
- It is written in Python
- Based upon that library there is a number of utilities
  - For case manipulation
  - Running simulations
  - Looking at the results
- All utilities start with pyFoam (so TAB-completion gives you an overview)
  - Each utility has an online help that is shown when using the --help-option
  - Additional information can be found
    - on https://openfoamwiki.net

What is swak4Foam

From `https://openfoamwiki.net/index.php/Contrib/swak4Foam`

> `swak4Foam` stands for SWiss Army Knife for Foam. Like that knife it rarely is the best tool for any given task, but sometimes it is more convenient to get it out of your pocket than going to the tool-shed to get the chain-saw.

- It is the result of the merge of
  - funkySetFields
  - groovyBC
  - simpleFunctionObjects

  and has grown since
- The goal of swak4Foam is to make the use of C++ unnecessary
  - Even for complex boundary conditions etc

The core of `swak4Foam`

- At its heart `swak4Foam` is a collection of parsers (subroutines that read a string and interpret it)
  - `"T-273.15"` is interpreted as "get the field `T` and subtract 273.15 from it (not changing the field, but creating a new one)"
- For expressions on `OpenFOAM`-types
  - fields
  - boundary fields
  - other (`faceSet`, `cellZone` etc)
- . . . and a bunch of utilities, function-objects and boundary conditions that are built on it
- `swak4foam` tries to reduce the need for throwaway C++ programs for case setup and postprocessing

Outline

Command line examples

- In the following presentation we will enter things on the command line. Short examples will be a single line (without output but a ">" to indicate *input*)

```
> ls $HOME
```

- Long examples will be a grey/white box
  - Input will be prefixed with a > and blue
  - Long lines will be broken up
    - A pair of <brk> and <cont> indicates that this is still the same line in the input/output
  - «snip» in the middle means: "There is more. But it is boring"

Long example

```
> this is an example for a very long command line that does not fit onto one line of the slide <brk>
      <cont> but we have to write it anyway
first line of output (short)
Second line of output which is too long for this slide but we got to read it in all its glory and<brk>
      <cont> will be probably broken
```

Work environment

- You will use two programs
  - A terminal
  - A text-editor
- For the text-editor you have the choice (these should be installed):
  - Emacs (king of text-editors)
  - VI
  - Kate with KDE
  - Gedit with Gnome
  - `nano`
  - `jedit`
  - ...

Getting onto the same page

- We need a machine with
  - OpenFOAM 7.0
    - but older versions work as well
    - and other forks like `foam-extend` or `v1906` (but with that the re-implementation of `lumpedWall` would be pointless)
  - swak4foam
  - PyFoam
  - Text editors: `emacs`, `vim`, `gedit`

Open a shell and set us up for work

```
> mkdir swakAndPyFoam
> cd swakAndPyFoam
> . ~/OpenFOAM/OpenFOAM-7/etc/bashrc
```

Docker image with pre-installed PyFoam and swak4Foam

- Docker is a technology to run pre-packed containers based on Linux
  - Can be run on Linux, Windoze and Mac OS X
  - Saves the work of installing requirements and compiling software
    - Only docker is needed (see `https://www.docker.com/`)
    - Image downloads may be rather big
- There is a container maintained by the author
  - Based on the official CFDDirect OpenFOAM 5.0 docker image
  - Most recent release of PyFoam
  - A development version of swak4Foam
- For installation instructions the README at `https://bitbucket.org/bgschaid/swak4foamandpyfoamdockerfile/src/default/`

Pulling the Docker-Image

Problems here:

- The image is over 3 Gig.
  - If you all do this now we might bring the network down
- You have to have Docker installed on your machine

Getting the script

```
> wget https://bit.ly/30Bvlgj -O swakPyFoam-of7.0
> chmod a+x swakPyFoam-of7.0
```

The actual URL for the script is `https://bitbucket.org/bgschaid/swak4foamandpyfoamdockerfile/raw/da4ec82e6fa0e6ff2544b734df37e774dc53c732/swakPyFoam-of7.0`

Starting the container

This will download the container the first time around

```
> ./swakPyFoam-of7.0
```

After that you're on a shell inside the container

Started docker container

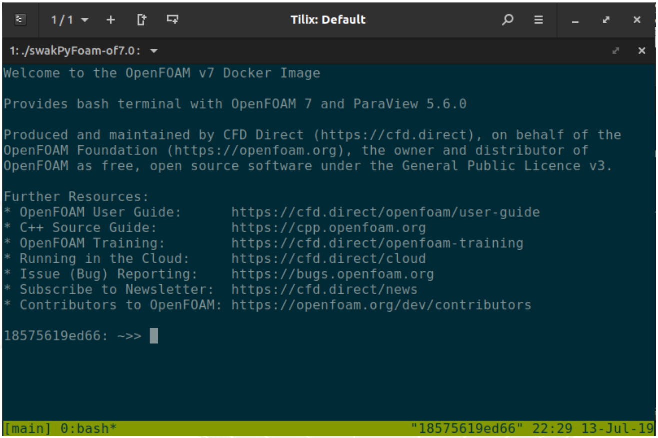

Figure: Docker container after start

Getting the Material

The 4 main stages of the presentation are archived in a tar

- But it should be possible to reproduce everything from the slides

Download stuff to the current directory

```
> wget https://openfoamwiki.net/images/a/a0/PyFoamSwak_Duisburg2019_Material.tar.gz
> tar xvzf PyFoamSwak_Duisburg2019_Material.tar.gz
01baseCase.tar.gz
02staticSetup.tar.gz
03simpleGroovyBC.tar.gz
04moreGroovyBC.tar.gz
```

Make sure PyFoam is working

- There is a utility that helps make sure that PyFoam is working
  - and gives valuable information for support

Getting the version

```
> pyFoamVersion.py
 Machine info: Darwin | bgs-cool-greybook | 16.6.0 | Darwin Kernel Version 16.6.0: Fri Apr
 14 16:21:16 PDT 2017; root:xnu-3789.60.24~6/RELEASE_X86_64 | x86_64 | i386

Python version: 3.5.3 (default, Apr 23 2017, 18:09:27)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)]

Python executable: /opt/local/bin/python

Python 3 is supported with PyFoam
PYTHONPATH: /Users/bgschaid/private_python:/Users/bgschaid/private_python:

Location of this utility: /Users/bgschaid/Development/OpenFOAM/Python/PyFoam/bin/pyFoamVe
rsion.py

Version 1706 (reported as number 1706 )Fork openfoam of the installed 27 versions:
                  extend-3.0 : /Users/bgschaid/foam/foam-extend-3.0
                  extend-3.1 : /Users/bgschaid/foam/foam-extend-3.1
<<snip>>
          openfoamplus-v1706 : /Users/bgschaid/OpenFOAM/OpenFOAM-v1706
          openfoamplus-v3.0+ : /Users/bgschaid/OpenFOAM/OpenFOAM-v3.0+

pyFoam-Version: 0.6.9-development

Path where PyFoam was found (PyFoam.__path__) is ['/Users/bgschaid/private_python/PyFoam']

Configuration search path: [('file', '/etc/pyFoam/pyfoamrc'), ('directory', '/etc/pyFoam/pyfoamrc.d'), ('file', '/<brk>
      <cont>Users/bgschaid/.pyFoam/pyfoamrc'), ('directory', '/Users/bgschaid/.pyFoam/pyfoamrc.d')]
Configuration files (used): ['/Users/bgschaid/.pyFoam/pyfoamrc', '/Users/bgschaid/.pyFoam/pyfoamrc.d/testit.cfg']

Installed libraries:
cython                         :  Yes    version: 0.25.2
cProfile                       :  Yes
docutils                       :  Yes    version: 0.13.1
Gnuplot                        :  No     Not a problem. Version from ThirdParty is used
hotshot                        :  No     Not a problem. Can't profile using this library
line_profiler                  :  No     Not a problem. Can't profile using this library
```

pyFoamVersion.py

- Information the utility gives
  - Machine
  - Used python
  - PYTHONPATH (where additional libraries are searched)
  - Information about the used PyFoam
    - Where configuration files are sought
  - Installed libraries relevant for PyFoam
    - With version if possible
- This information helps diagnosing problems
  - Copy this output when reporting problems that might be associated with the installation

Make sure swak4Foam is installed

- Call the most popular utility of swak4Foam
  - swakVersion reported below the usual header

Provoking an error

```
> funkySetFields
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1612+                                |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : v1612+-25c3270ac2a3
Exec   : funkySetFields
Date   : Jul 09 2017
Time   : 21:03:21
Host   : "bgs-cool-greybook"
PID    : 59774
Case   : /Volumes/Foam/LatexDocs/Vortraege/Exceter2017/Exceter2017SwakPyFoam/Vortrag
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using timeStampMaster (fileModificationSkew 10)
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
swakVersion: 0.4.1 (Release date: 2017-05-31)
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


--> FOAM FATAL ERROR:
funkySetFields: time/latestTime option is required


    From function main()
    in file funkySetFields.C at line 759.

FOAM exiting
```

Outline

Outline

Getting the base case

- pyFoamCloneCase.py only copies the parts of a case that are necessary to start it
  - system, constant, 0
- We move 0 to 0.org to avoid overwriting it
- PyFoamHistory records what is done to the case with PyFoam
  - Handy for "What command did I use 3 weeks ago to prepare this?"
- We don't need the Allrun / Allclean scripts
- PyFoam creates a .foam-file so that we can open the case in ParaView

Using our first PyFoam utility

```
> pyFoamCloneCase.py $FOAM_TUTORIALS/heatTransfer/buoyantPimpleFoam/hotRoom 01baseCase
 PyFoam WARNING on line 117 of file /Users/bgschaid/private_python/PyFoam/Applications/<brk>
      <cont>CloneCase.py : Directory  does not exist. Creating
> cd 01baseCase
> ls
0               Allclean        PyFoamHistory   system
01baseCase.foam Allrun          constant
> rm All*
> mv 0 0.org
> cat PyFoamHistory
Fri Jul 14 00:14:40 2017 by bgschaid in bgs-cool-greybook :Application: pyFoamCloneCase.py <brk>
     <cont>/Users/bgschaid/OpenFOAM/OpenFOAM-4.1/tutorials/heatTransfer/buoyantPimpleFoam/<brk>
     <cont>hotRoom 01baseCase  | with cwd /path/to/the/case  |  Cloned to 01baseCase
```

Preparing

pyFoamPrepareCase.py is a utility to set up cases in a reproducible way

First setup

```
> pyFoamPrepareCase.py .
Looking for template values .

Used values

              Name - Value
----------------------------------------
          caseName - "01baseCase"
          casePath - "/path/to/the/case/01baseCase"
          foamFork - openfoam
       foamVersion - 4.1
numberOfProcessors - 1

No script ./derivedParameters.py for derived values
Clearing .
 PyFoam WARNING on line 642 of file /Users/bgschaid/private_python/PyFoam/RunDictionary/<brk>
      <cont>SolutionDirectory.py : The first timestep in /path/to/the/case/01baseCase  is  <brk>
      <cont>None not a number. Doing nothing
Writing parameters to ./PyFoamPrepareCaseParameters
Writing report to ./PyFoamPrepareCaseParameters.rst
Found 0.org. Clearing 0
No 0-directory
...
```

What `pyFoamPrepareCase.py` does

- It does more. But in our case it
  1. Removes old timesteps
  2. Copies `0.org` to `0`
  3. runs `blockMesh`
     - because it found a `blockMeshDict`
  4. runs `setFields`
- There is a full presentation about this utility
  - Does a lot more:
    - Create files from templates
    - Executes scripts to set up the case
    - ...

Running

This is the most-used utility in PyFoam

Starting the simulation

```
> pyFoamPlotRunner.py --clear --progress --auto --hardcopy --prefix=firstRun auto
Clearing out old timesteps ....
 Warning in /Users/bgschaid/Development/OpenFOAM/Python/PyFoam/bin/pyFoamPlotRunner.py : <brk>
      <cont>Replacing solver 'auto' with buoyantPimpleFoam in arguments
t =      232
```

Some time later

```
t =      2000
> ls
0                                      PyFoamPrepareCaseParameters
0.org                                  PyFoamPrepareCaseParameters.rst
01baseCase.foam                        PyFoamRunner.buoyantPimpleFoam.analyzed
1000                                   PyFoamRunner.buoyantPimpleFoam.logfile
1200                                   PyFoamServer.info
1400                                   PyFoamState.CurrentTime
1600                                   PyFoamState.LastOutputSeen
1800                                   PyFoamState.LogDir
200                                    PyFoamState.StartedAt
2000                                   PyFoamState.TheState
400                                    constant
600                                    firstRun.cont.png
800                                    firstRun.linear.png
Gnuplotting.analyzed                   hotRoomMoving.foam
PyFoam.blockMesh.logfile               PyFoam.setFields.logfile
system                                 PyFoamHistory
```

What pyFoamPlotRunner.py does

- Executes a solver
- Captures the output
  - Writes it to a logfile
    - Starts with `PyFoamRunner` and ends with `logfile`
  - Analyzes it and plots the results
- The options we used are

–clear: Remove old simulation results

–progress: Swallow the output and only print the time

–auto: if we find `processor*`-directories run the case in parallel. If not: run single processor

–hardcopy , –prefix: In the end create pictures of the plots. Start their names with `firstRun`

Residuals plot

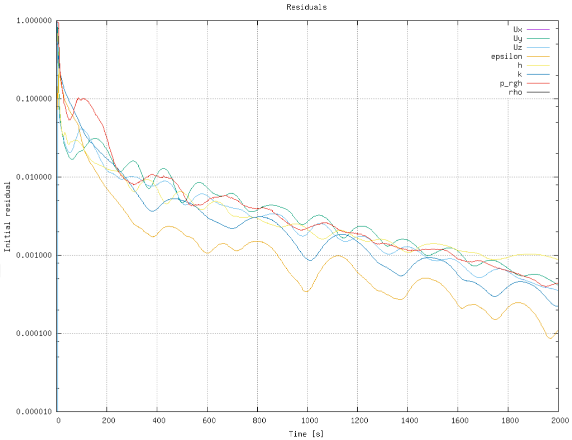

Figure: Automatic plot of the initial residuals

Continuity plot

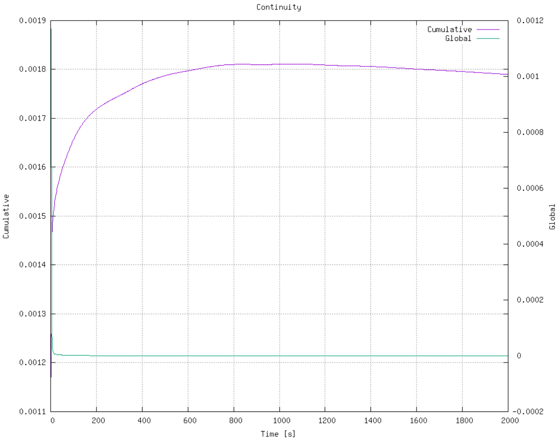

Figure: Automatic plot of the continuity

Watching

- The utility pyFoamPlotWatcher.py takes a file and interprets it as the output of an OpenFOAM-run
  - Assumes that the file is not "finished" and updates the plots when lines are added
- Options are similar to the PlotRunner
  - --with-all adds some more plots

Replaying the plots

```
> pyFoamPlotWatcher.py --with-all --hardcopy --prefix=firstRunWatch PyFoamRunner.<brk>
      <cont>buoyantPimpleFoam.logfile
<snip>
diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
time step continuity errors : sum local = 1.32491e-09, global = -1.69522e-11, cumulative = <brk>
      <cont>0.00179062
DILUPBiCG:  Solving for epsilon, Initial residual = 0.000109711, Final residual = 1.21588e<brk>
      <cont>-07, No Iterations 1
DILUPBiCG:  Solving for k, Initial residual = 0.00022317, Final residual = 4.67542e-07, No <brk>
      <cont>Iterations 1
ExecutionTime = 31.74 s  ClockTime = 55 s

End
^C
Watcher: Keyboard interrupt
> ls *.png
firstRun.cont.png          firstRunWatch.courant.png   firstRunWatch.linear.png
firstRun.linear.png        firstRunWatch.execution.png
firstRunWatch.cont.png     firstRunWatch.iter.png
```

Number of iterations

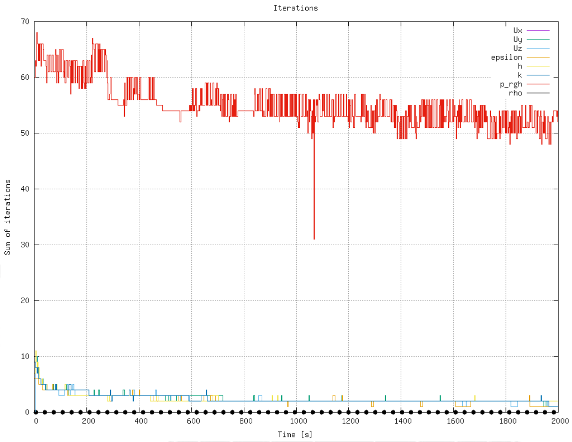

Figure: Automatic plot of iterations of the linear solver

Execution time

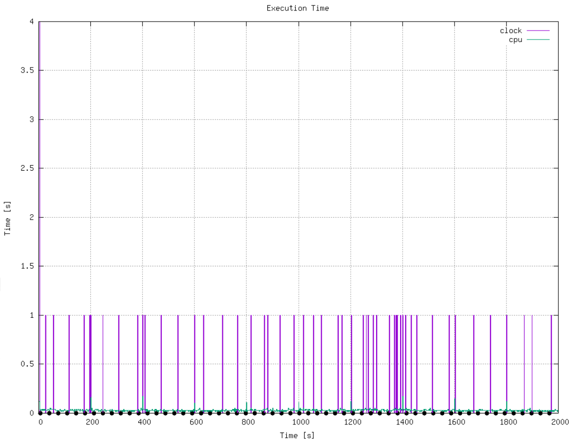

Figure: The time each timestep takes (jumps because of resolution of the output)

Courant number

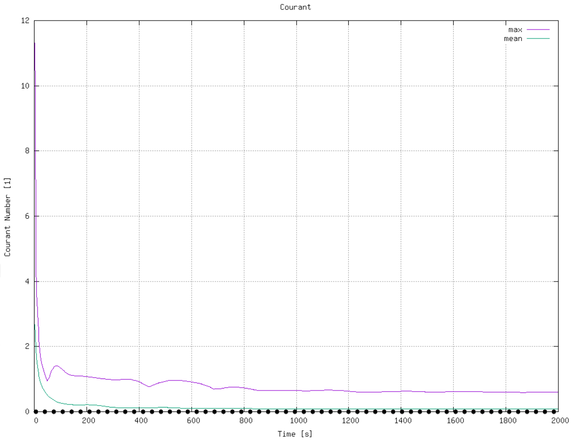

Figure: Courant numbers calculated by OpenFOAM

Outline

State files in ParaView

- Great time-saving feature of ParaView
  - Which now (== the last few years) works quite stable
- The way to work with it
  1. Do a complicated visualization
  2. Save it with `Save State`
  3. Close Paraview
  4. Copy state-file to another case
  5. Open Paraview
  6. Press `Load state` and select state-file
  7. Paraview is confused and asks for the case
  8. Do the same visualization with another case
- Saves a lot of time
  - But it can be even easier

Example of Paraview state

- Create a visualization that you like
  - Important : A Text source with the content %(casename)s

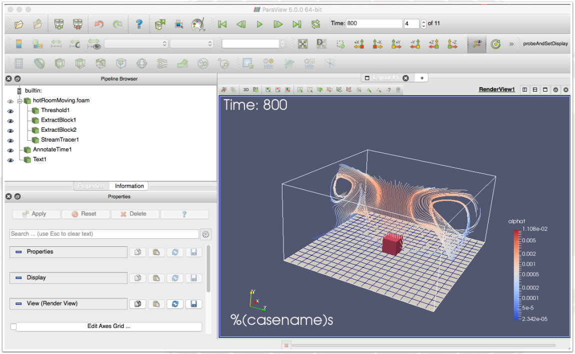

Figure: How Paraview looks before we save the state

pyFoamPVSnapshot.py

- Utility in pyFoam that needs three informations
  1. A state-file
  2. The case
  3. One or more times
- In return it does:
  1. Create a copy of the state-file
  2. Manipulate it to point to the case
  3. Load into a GUI-less version of Paraview (pvpython)
  4. Write pictures
- Can do a few other things
- This allows quickly creating reference pictures for similar cases
  - Which look exactly the same

No Paraview

- Now we can create pictures without using the mouse
- `--state` is the state-file we created
- `--time` and `--latest` specify which times to snapshot
- The `.` says "this directory/case"

Creating the pictures

```
> pyFoamPVSnapshot.py . --state=hotWithStreamlines.pvsm --time=200 --latest
Executing PVSnapshot with pvpython trough a proxy-script options:
Warning in /var/folders/h7/3nw065_955d1zm30_bjn384h0000gr/T/pyFoamPVSnapshot_du5hxr1z.py : <brk>
      <cont>Setting decomposed type to auto : Decomposed/Reconstruced correctly set. Nothing<brk>
      <cont> changed
PyFoam WARNING on line 110 of file /Users/bgschaid/private_python/PyFoam/Paraview/<brk>
      <cont>ServermanagerWrapper.py : Can't find expected plugin 'libPOpenFOAMReaderPlugin' <brk>
      <cont>assuming that correct reader is compiled in. Wish me luck Warning in /var/<brk>
      <cont>folders/h7/3nw065_955d1zm30_bjn384h0000gr/T/pyFoamPVSnapshot_du5hxr1z.py : <brk>
      <cont>Trying offscreen rendering. If writing the file fails with a segmentation fault <brk>
      <cont>try --no-offscreen-rendering
Snapshot  1  for t= 200 View 0 png
Snapshot  10  for t= 2000 View 0 png
Warning in /var/folders/h7/3nw065_955d1zm30_bjn384h0000gr/T/pyFoamPVSnapshot_du5hxr1z.py : <brk>
      <cont>Removing pseudo-data-file /path/to/the/case/01baseCase/01baseCase.OpenFOAM
> ls Snap*
Snapshot_01baseCase_00001_t=200_hotWithStreamlines.png
Snapshot_01baseCase_00010_t=2000_hotWithStreamlines.png
```

Note : For some reason this doesn't work in the Docker container

Simulation at start

Note: `%(casename)s` has been replaced with the name of the case

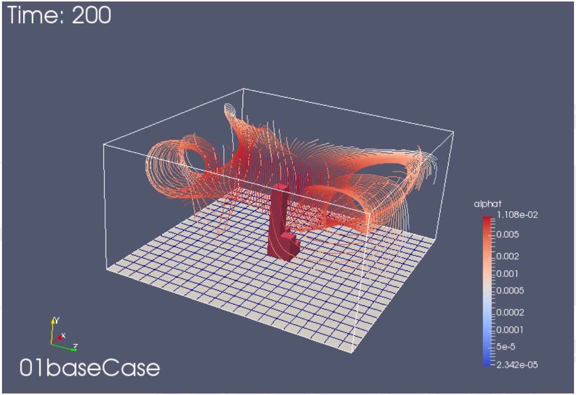

Figure: First written time-step

Almost steady state

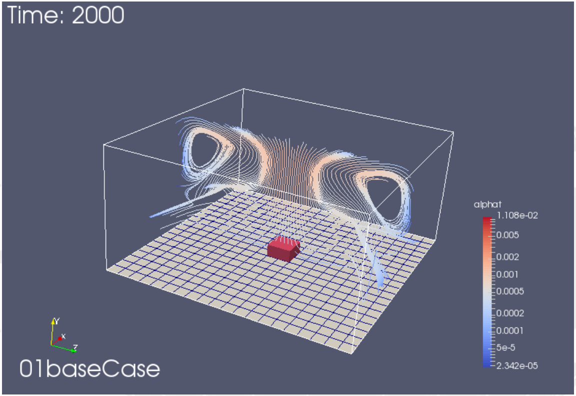

Figure: Flow has developed

Give me the numbers

- Sometimes one opens `Paraview` just to see the ranges of the variables
  - The numbers the post-processor shows are not necessarily the numbers OpenFOAM uses
- There is a utility to quickly check that

Getting numbers

```
> fieldReport -time 2000 T
<snip>
Time = 2000

 Reading Field T of type volScalarField

Internal field:
swak4Foam: Allocating new repository for sampledMeshes
swak4Foam: Allocating new repository for sampledGlobalVariables
Size | Weight Sum                4000 |          500
Range (min-max)               300.458 |      300.941
Average | weighted            300.532 |      300.532
Sum | weighted            1.20213e+06 |       150266
Median | weighted             300.535 |      300.535

End
```

Not all the numbers make sense for all fields

Numbers from `fieldReport`

"Weight" is the cells volumes

Size Number of cells

Weight Sum Total volume of the case

Range The .... range

Average average of all cells (each cell has weight 1)

- **weighted** average weighted by the cell volume

Sum Value in all cells added (usually makes no sense)

- **weighted** basically the integral (only makes sense for extensive values)

Median The value for which 50% of the cells have a smaller value (more stable than `Average`)

- This is used quite often in `swak4Foam`
- Generalization is `quantile`: `quantile0.5` is the same as `median`
- `fieldReport` can report these too: see `-help`

More numbers

- Utility can report patches separately
- Write to csv-files to be analyzed elsewhere
  - entity allows separating the data

Drowning in data

```
> fieldReport -time 0: -doBoundary -csvName numbers T
<snip>

Patch field: fixedWalls
Size | Weight Sum                800 |          200
Range (min-max)               300.462 |       300.55
Average | weighted            300.529 |      300.529
Sum | weighted                 240424 |      60105.9
Median | weighted             300.534 |      300.534

End
> ls *csv
numbers_T_region0.csv
> cat numbers_T_region0.csv
time,entity,size,weight_sum,minimum,maximum,average,average_weighted,sum,sum_weighted,<brk>
      <cont>median,median_weighted
0,internalField,4000,500,300,300,300,300,1.2e+06,150000,300,300
0,patch floor,400,100,300,600,303,303,121200,30300,300.505,300.505
0,patch ceiling,400,100,300,300,300,300,120000,30000,300,300
0,patch fixedWalls,800,200,300,300,300,300,240000,60000,300,300
200,internalField,4000,500,300.405,302.24,300.511,300.511,1.20204e<brk>
      <cont>+06,150255,300.499,300.499
200,patch floor,400,100,300,600,303,303,121200,30300,300.505,300.505
200,patch ceiling,400,100,300,300,300,300,120000,30000,300,300
```

Throwing all away

- pyFoamClearCase.py does the same thing as the --clear-option of the Runner
  - Throws non-essential stuff away
  - --keep-last means "and keep the final result"

```
> pyFoamClearCase.py --verbose-clear --keep-last .
Clearing /path/to/the/case/01baseCase/200
Clearing /path/to/the/case/01baseCase/400
<snip>
Clearing /path/to/the/case/01baseCase/1600
Clearing /path/to/the/case/01baseCase/1800
Clearing /path/to/the/case/01baseCase/PyFoam.blockMesh.logfile
Clearing /path/to/the/case/01baseCase/PyFoam.setFields.logfile
Clearing /path/to/the/case/01baseCase/PyFoamPrepareCaseParameters
```

Packing the case with pyFoamPackCase.py

- Similar to pyFoamCloneCase.py
  - Knows "what is important"
  - But instead creates an archive file
- The state until here has been packed with
  - The state file is added

```
pyFoamPackCase.py 01baseCase --add=hotWithStreamlines.pvsm
```

- This file is found as 01baseCase.tar.gz in the material
- if things happened too fast to follow

Outline

Outline

funkySetFields

- This utility is the oldest part of `swak4Foam`
  - Existed looong before `swak4Foam`
- The idea is "specify an expression and the utility creates a field with that value"
  - Or modify an existing field
- Most important options are

  `-time` and `-latestTime` Which times to use

  `-field` Name of the field to write

  `-create` (optional) Create a new field

  `-expression` The expression that should be evaluated

  `-condition` (optional) only modify cells where this logical expression is `true`

For our non-metric friends

It is hard enough to think "Is 300 K warm for a room?" if you're used to Celsius. But if you're used to Fahrenheit . . . .

Calculating the room temperature

```
> funkySetFields -time 0: -create -field TFahrenheit -expression "T*(9/5)-459.67"
<snip>
Time = 2000
 Using command-line options

 Creating field TFahrenheit

 Putting "T*(9/5)-459.67" into field TFahrenheit at t = "2000" if condition "true" is true

 Setting 4000 of 4000 cells
 Writing to "TFahrenheit"
End
> fieldReport -time 0:  TFahrenheit
<snip>
Time = 2000

 Reading Field TFahrenheit of type volScalarField

Internal field:
Size | Weight Sum                4000 |         500
Range (min-max)               81.1544 |     82.0238
Average | weighted            81.2876 |     81.2876
Sum | weighted                 325150 |     40643.8
Median | weighted             81.2919 |     81.2919

End
```

Old way of setting the boundaries

This is how the original case set the boundary value

setFieldsDict

```
defaultFieldValues
(
    volScalarFieldValue T 300
);

regions
(
    // Set patch values (using ==)
    boxToFace
    {
        box (4.5 -1000 4.5) (5.5 1e-5 5.5);

        fieldValues
        (
            volScalarFieldValue T 600
        );
    }
);
```

Doing it our own way

1. Remove the old file

```
rm system/setFieldsDict
```

1. Setting up the case

```
pyFoamPrepareCase.py .
```

1. Run funkySetFields:

Shell

```
> funkySetFields -time 0 -keepPatches -valuePatches "floor" -field T -expression "600" -<brk>
      <cont>condition "(pos().x>4.5 && pos().x<5.5 && pos().z>4.5 && pos().z<5.5)"
<snip>
Time = 0
 Using command-line options

 Modifying field T of type volScalarField

 Putting "600" into field T at t = "0" if condition "(pos().x>4.5 && pos().x<5.5 && pos().z<brk>
      <cont>>4.5 && pos().z<5.5)" is true
 Keeping patches unaltered

 Setting 40 of 4000 cells
 Writing to "T"
End
```

Explanation

- If you never programmed C/C++/Java:
  - `&&` means "logical and"
- `pos()` is the position of the cell center
  - `.x` is the x-component
- `-keepPatches` means "keep that patches that we found in the original file"
  - Note: we didn't use `-create`
- `-valuePatches` is a list with patches were the value from the cells near to the patch are used for the patch faces
  - Otherwise `zeroGradient` is default for patches

Expression syntax

- The syntax of swak4Foam expressions is based on the syntax OpenFOAM uses in its programs
  - Which in turn is C++
    - The usual operator precedence (multiplication before addition etc) applies
  - "Special" operators like & for the inner product and ^ are the same as in "OpenFOAM C++"
- There is a number of builtin-functions based on the regular OpenFOAM-functionality
  - This includes differential operators like `div` or `snGrad`
    - But only the explicit variation
- Expressions give the same results in parallel
  - No need to change anything on the user side
  - This includes `min`, `max` and `average`
- Not all functions will be explained here
  - For a complete list look at the *Incomplete reference guide*

Column of fire

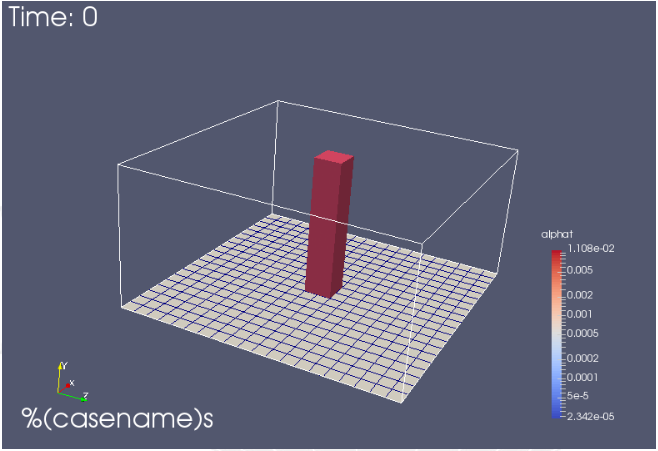

Figure: Initial condition as seen in Paraview

Clearing it

- We don't want the "column of fire" as an initial condition
  - But the patches should be left intact
  - because of valuePatches the floor has the desired values

Removing the inner values

```
> funkySetFields -time 0 -keepPatches  -field T -expression "300"
<snip>
Time = 0
 Using command-line options

 Modifying field T of type volScalarField

 Putting "300" into field T at t = "0" if condition "true" is true
 Keeping patches unaltered

 Setting 4000 of 4000 cells
 Writing to "T"
End
> pyFoamPlotRunner.py --clear --progress auto
```

Calling funkySetFields automatically

- Calling this funkySetFields by hand every time we change the mesh is tedious
- pyFoamPrepareCase.py can do this for us
  - A script caseSetup.sh is called after the mesh creation
- Copy the commands from the terminal to the script:

caseSetup.sh

```
#! /bin/sh
funkySetFields -time 0 -keepPatches -valuePatches "floor" -field T -expression "600" -<brk>
      <cont> condition "(pos().x>4.5␣&&␣pos().x<5.5␣&&␣pos().z>4.5␣&&␣pos().z<5.5)"
funkySetFields -time 0 -keepPatches  -field T -expression "300"
```

Outline

Adding function objects

- *function objects* are small programs that are executed at the end of every time-step
  - OpenFOAM has a lot of them
  - Most of the functionality in `swak4Foam` is in function objects
- They have to be loaded at run-time
  - By adding the library in to the `libs` list in `controlDict`
- Function objects are added to the `functions`-dictionary in `controlDict`
  - Need a unique name
  - Only required parameter is the `type`
    - Everything else depends on the `type`

system/controlDict

```
libs (
    "libsimpleSwakFunctionObjects.so"
);
```

Evaluating the temperature

At first we want to get the statistics of the temperature at every time-step

system/controlDict

```
functions {
    temperatures {
        type swakExpression;
        valueType internalField;
        verbose true;
        expression "T";
        accumulations (
            min
            weightedQuantile0.1
            weightedAverage
            weightedQuantile0.9
            max
        );
    }
}
```

swakExpression

- One of the most general function objects in `swak4Foam`
  - Evaluates an `expression` on a part of the mesh (cell zone, patch, ...)
  - Which part is specified by `valueType`
    - `internalMesh` means "in the cells"
  - `verbose` means "write to the console"
    - Otherwise only a file in `postProcessing` is written
  - `accumulations` is a list of ... accumulations
    - *Accumulation* here means "a method to take many numbers and condense them into one number"
    - A list of all the accumulations can be found in the *Incomplete Reference Guide* that comes with the swak-sources

Running with Evaluation

- How the output looks like will be important in the next step
  - Copy the line with the temperature to later paste it into the text editor
    - This avoids typos

Example output

```
> pyFoamRunner.py --clear auto
<snip>
DILUPBiCG:  Solving for k, Initial residual = 0.055352, Final residual = 1.78789e-09, No <brk>
     <cont> Iterations 5
ExecutionTime = 1.7 s  ClockTime = 4 s

Expression temperatures :  min=300.375 weightedQuantile0.1=300.425 weightedAverage=300.487 <brk>
     <cont> weightedQuantile0.9=300.526 max=302.99
Courant Number mean: 0.314849 max: 1.40162
Time = 86

diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
PIMPLE: iteration 1
<snip>
> ls postProcessing/swakExpression_temperatures/0
temperatures
```

Getting PyFoam to recognize what swak4foam calculated

- We'd like to have plots of the temperature
- The way this works is
  1. swak4foam writes the numbers to the console
  2. PyFoam grabs that output
  3. Analyzes it
  4. If it finds things it recognizes it collects them
  5. And plots them
- We've got to tell PyFoam about the stuff it should recognize
  - For this we give it a `customRegexp`-file
    - In that file we need *regular expressions*

Regular expressions

- Regular expressions are very popular for analyzing textual data (pattern matching)
  - For instance in OpenFOAM for flexible boundary conditions
  - Python comes with a library for analyzing them
  - There are slightly different dialects
    - For instance there are slight differences between the regular expressions of Python and OpenFOAM
    - But in 90% of all cases they behave the same
- The following slide gives a quick glance
  - Usually you won't need much more for PyFoam
- There is a number of cool "regular expression tester" (enter that in Google) applications on the web
  - One example: `http://regex101.com`

Regular expressions in 3 minutes

1. Most characters match only themself
   - For instance 'ab' matches only the string "ab"
2. The dot ('.') matches any character except a newline
   - Pattern 'a..a' matches (among others) "abba", "aBBa", "ax!a"
3. The plus '+' matches the character/pattern before it 1 or more times
   - 'a.+a' matches "aba", "abbbba" but not "aa"
4. '*' is like '+' but allows no match too
   - 'a.*a' matches "aba", "abbbba" and also "aa"
5. Parenthesis '()' group characters together. Patterns are numbered. They receive the number by the opening '('
   - 'a((b+)a)' would match "abba" with group 1 being "bba" and group 2 "bb"
6. To match a special character like '+-().|' prefix it with a '\'
   - To match "(aa)" you've got to write '\(aa\)'
   - Other special characters that occur frequently in OpenFOAM-output are '[]\{\}'

The customRegexp-file

- If a file `customRegexp` is found in the case by a `Plot`-utility it is read
- It is in OpenFOAM-format:
  - a dictionary
  - all entries are dictionaries too
- The name of the entry is used to identify the data (for instance during writing)
- Most frequent entry in the dictionaries are:

  expr This is required. A regular expression that a line must match. All groups (enclosed by '()') are interpreted as data and plotted

  theTitle String with the title of the plot

  titles List of words/strings. The names that the data items will get in the legend
- `customRegexp` is important enough for PyFoam to be automatically cloned by `pyFoamCloneCase.py`

PyFoam reads the temperature

- Paste the line you copied before into the customRegexp-file
  - Build the rest around it
    - If there are special characters in the output put a backslash before it
    - Replace the numbers you want with (.+). If you don't need them replace with .+ (no ())
  - Because just one forgotten (or extra) space will make the expression not match the output

customRegexp

```
// -*- c++ -*-

temperature {
    theTitle "Temperature";
    ylabel "T[K]";
    expr "Expression␣temperatures␣:␣␣min=(.+)␣weightedQuantile0.1=(.+)␣weightedAverage=(.+)<brk>
          <cont>␣weightedQuantile0.9=(.+)␣max=(.+)";
    titles (
        min
        "10␣%"
        average
        "90␣%"
        max
    );
}
```

Remark: First line is only for Emacs-users

The temperature plot

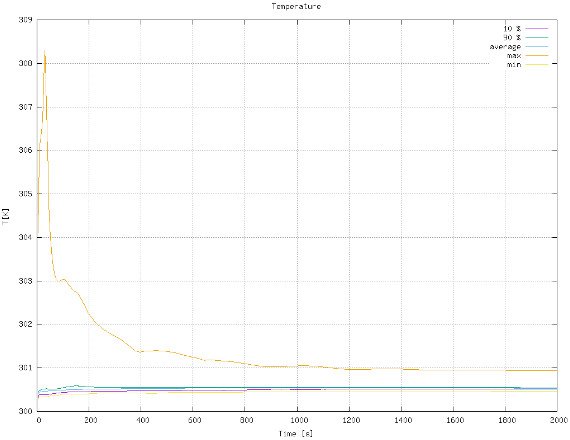

Figure: T by `pyFoamPlotRunner`

Adding patch temperatures

- Now we want to know the temperatures on the patches
- `patchExpression` is a specialized version of `swakExpression`
  - Doesn't need `valueType`
  - But a list `patches` with the patch names
- The function `internalField` doesn't use the patch-face values but the next cells
  - Much better in this case

In functions in system/controlDict

```
wallTemperatures {
    $temperatures;
    type patchExpression;
    patches (
        floor
        ceiling
        fixedWalls
    );
}
wallTemperaturesInternal {
    $wallTemperatures;
    expression "internalField(T)";
}
```

Patch output

More output

```
ExecutionTime = 2.54 s  ClockTime = 5 s

Expression temperatures :  min=300.404 weightedQuantile0.1=300.446 weightedAverage=300.499 <brk>
      <cont>weightedQuantile0.9=300.574 max=302.878
Expression wallTemperatures on fixedWalls:  min=300.404 weightedQuantile0.1=300.436 <brk>
      <cont>weightedAverage=300.46 weightedQuantile0.9=300.485 max=300.499
Expression wallTemperatures on floor:  min=300 weightedQuantile0.1=300.101 weightedAverage<brk>
      <cont>=303 weightedQuantile0.9=300.909 max=600
Expression wallTemperatures on ceiling:  min=300 weightedQuantile0.1=300 weightedAverage<brk>
      <cont>=300 weightedQuantile0.9=300 max=300
Expression wallTemperaturesInternal on fixedWalls:  min=300.404 weightedQuantile0.1=300.436<brk>
      <cont> weightedAverage=300.46 weightedQuantile0.9=300.485 max=300.499
Expression wallTemperaturesInternal on floor:  min=300.424 weightedQuantile0.1=300.445 <brk>
      <cont>weightedAverage=300.492 weightedQuantile0.9=300.505 max=302.878
Expression wallTemperaturesInternal on ceiling:  min=300.404 weightedQuantile0.1=300.422 <brk>
      <cont>weightedAverage=300.516 weightedQuantile0.9=300.719 max=300.901
Courant Number mean: 0.231081 max: 1.171
Time = 128
```

There is a lot information here. But it is hard to read

All walls in one plot

- Here we use a `dynamic` plot
  - "Dynamically generate data sets from a name"
- Name is taken from the `idNr`-th regular expression group

customRegexp

```
wallInternalTemperatures {
    theTitle "Temperature near the wall";
    type dynamic;
    idNr 1;
    expr "Expression wallTemperaturesInternal on (.+):  min=(.+) weightedQuantile0.1=(.+) <brk>
      <cont>weightedAverage=(.+) weightedQuantile0.9=(.+) max=(.+)";
    titles (
        min
        "10%"
        average
        "90%"
        max
    );
}
```

Plotting the wall temperatures

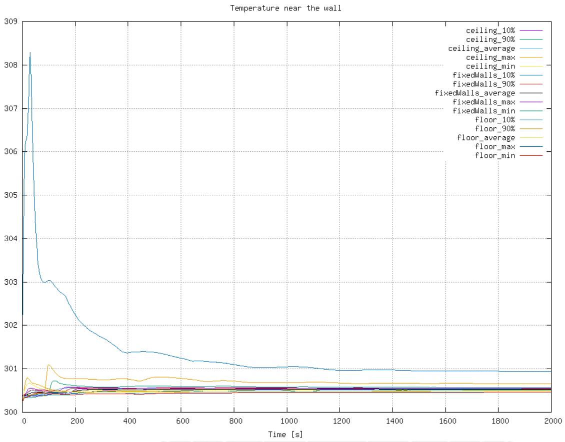

Figure: The temperatures near the wall

Outline

Fahrenheit while we go

- The expressionField function object is like "funkySetFields during the calculation"
  - But it is in another library

Additional line in controlDict

```
libs (
    "libsimpleSwakFunctionObjects.so"
    "libswakFunctionObjects.so"
);
```

functions in controlDict

```
addFahrenheit {
    type expressionField;
    autowrite true;
    expression "T*(9/5)-459.67";
    fieldName TFahrenheit;
}
```

autowrite Write at output times

Running and checking

Checking if that field was created

```
> pyFoamRunner.py --clear --auto --progress
t =       2000
> ls 2000
T.gz          U.gz          epsilon.gz    nut.gz        p_rgh.gz      swak4Foam
TFahrenheit.gz alphat.gz     k.gz          p.gz          phi.gz        uniform
> fieldReport -latestTime TFahrenheit
<snip>
Time = 2000

 Reading Field TFahrenheit of type volScalarField

Internal field:
swak4Foam: Allocating new repository for sampledMeshes
swak4Foam: Allocating new repository for sampledGlobalVariables
Size | Weight Sum                  4000 |           500
Range (min-max)                 81.1543 |        82.024
Average | weighted              81.2876 |       81.2876
Sum | weighted                   325151 |       40643.8
Median | weighted                81.292 |        81.292

End
```

Dictionary mode of `funkySetFields`

- Until now we used `FSF` in command-line mode
  - Everything is specified on the command line
- In dictionary-mode only the time is specified on the command line
  - Everything else in a dictionary
  - A list of `expressions`
  - Each `expression` is a named dictionary
    - Dictionary entries correspond to command line options
    - And there are more
- Advantages of dictionary mode:
  - More than one evaluation possible
  - More flexibility

The `variables`-list

- Almost everywhere where we have a `expression` we can specify such a list
- One expression needs these 4 components:
  1. Variable name
  2. =
  3. Expression
  4. ;
- The expression is evaluated and stored under the name
- Purpose: make expressions more readable by breaking them into part
  - Disadvantage: needs memory
- `variables` are evaluated every time the expression is evaluated

Normalizing by the length

- Here we make the velocity "dimensionless" by dividing it with the biggest length of the geometry
  - For calculating that length we use point locations `pts()` (not the cell locations)
- `max` is used in two ways here:
  - maximum of a field (gives one homogeneous field)
  - maximum of two values (may give a different value in every cell)

system/funkySetFields.nodimVel

```
expressions (
    velWithoutDimensions {
        field UDimless;
        create true;
        expression "U/LMax";
        variables (
            "xLen=max(pts().x)-min(pts().x);"
            "yLen=max(pts().y)-min(pts().y);"
            "zLen=max(pts().z)-min(pts().z);"
            "LMaxP=max(xLen,max(yLen,zLen));"
            "LMax=interpolateToCell(LMaxP);"
        );
        dimensions [0 0 -1 0 0 0 0];
    }
);
```

Native versus secondary structure

- For the `internalField` the value in the cells is the *native* value
- The values at the points is a secondary value
- `swak4Foam` does not automatically interpolate between them
  - For that functions like `interpolateToCell` have to be used
  - Constants like `1` are always native values
- For a list of native/secondary structures see the *Incomplete Reference guide* that comes with the sources

Just checking

What value should this give in the whole field? (theoretically)

```
> funkySetFields -time 0: -dictExt dimlessVel
<snip>
Time = 0
 Using funkySetFieldsDict

Part: velWithoutDimensions
 Creating field UDimless

 Putting "U/LMax" into field UDimless at t = "0" if condition "true" is true

swak4Foam: Allocating new repository for sampledMeshes
swak4Foam: Allocating new repository for sampledGlobalVariables
 Setting 4000 of 4000 cells
 Writing to "UDimless"
<snip>
> funkySetFields -time 2000 -expression "mag(U)/(1e-10+mag(UDimless))" -field relU -create
> fieldReport -time 2000 relU
```

Note: the 10^{-10} is there to avoid "divison by zero" errors

curse d output

- Some PyFoam-utilities have an option `--curses`
  - This uses the `curses` library for "GUIs" on the terminal
- Enhances the regular OpenFOAM-output with
  - A title line with PyFoam-utility, arguments and OpenFOAM-version
  - Another title line with the solver and the case
  - A footer line with
    - the number of lines in the log-file
    - time range of the simulation
    - number of timesteps so far
  - a line with the current time and progress information
  - On the bottom there is a *progress bar* with
    - current timestep number and estimated number of total timesteps
    - current wall-clock duration of the run and estimated time till simulation end
    - number of timesteps per second or seconds per timestep (whichever is bigger that 1)
  - Between that is the regular output
    - Lines that PyFoam got information from are colored differently
    - "Match groups" of the regular expressions are "enhanced"

"I put a spell on the output"

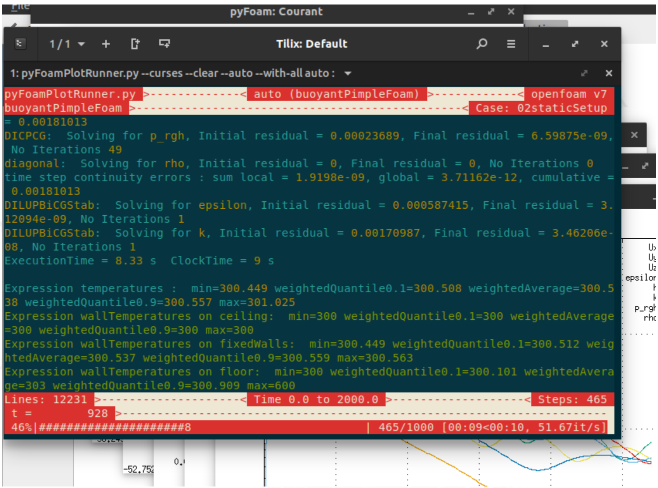

Figure: Output with --curses added to pyFoamPlotRunner

This case

All we've done so far can be found in `02staticSetup.tar.gz` in the material

Outline

Outline

Prepare

- Now instead of the static boundary condition we want a dynamic one
  - That's why we're removing `caseSetup.sh`
- The `pyFoamListCases.py`-utility is like a "`ls` for OpenFOAM-cases"

Checking our cases

```
> cd ..
> pyFoamCloneCase.py 02staticSetup 03simpleGroovyBC
> pyFoamListCases.py .
steps: 100%|*****************************************| 3/3 [00:00<00:00, 10.99entries/s]
                    mtime |           hostname | first - last (nrSteps) nowTime s    state | <brk>
                          <cont>             solver | name
-----------------------------------------------------------------------------------------------
       <cont>
Fri Jul 14 16:44:54 2017 | bgs-cool-greybook |     0 - 2000 (      2)  2000.0 s Finished | <brk>
       <cont>                  None | ./01baseCase
Sun Jul 16 09:27:39 2017 | bgs-cool-greybook |     0 - 2000 (     11)  2000.0 s Finished | <brk>
       <cont>buoyantPimpleFoam | ./02staticSetup
Sun Jul 16 13:26:16 2017 |          <no file> |  None - None (      0)    None s          | <brk>
       <cont>        none found | ./03simpleGroovyBC
> cd 03simpleGroovyBC
> rm caseSetup.sh
```

Adding another library

- We add the library for the dynamic boundary condition
- Also set the simulation time to a full hour

system/controlDict

```
libs (
    "libsimpleSwakFunctionObjects.so"
    "libswakFunctionObjects.so"
    "libgroovyBC.so"
);
```

and also

```
endTime 3600;
```

groovyBC

- This is the second oldest part of `swak4Foam`
  - The "fusion" of this and `FSF` became `swak4Faom`
- It is basically a `mixed` boundary condition where everything can be evaluated
  - **valueExpression** an expression describing the value
  - **gradientExpression** the gradient (this is optional)
  - **fractionExpression** whether the value is used (1) or the gradient (0). If unset a constant 1 is assumed
- Today we'll only use `valueExpression`

Setting a round heater

- Here we specify a moving heater
  - Heater is a circle with diameter: 1.5 m
  - The center moves on a circle with a radius of 1.5 m
  - Needs an hour to move around

0.org/T

```
floor
{
    type            groovyBC;
    value           uniform 300;
    variables (
        "center=vector(5,0,5);"
        "radiusFire=0.75;"
        "radiusCircle=1.5;"
        "radiant=2*pi*time()/3600;"
        "middle=center+radiusCircle*vector(sin(radiant),0,cos(radiant));"
        "tHigh=600;"
        "tLow=300;"
    );
    valueExpression "mag(pos()-middle)<radiusFire ? tHigh : tLow";
}
```

The conditional operator

- The ? : operator is known to those who ever programmed a language with a C-like syntax
- This is basically a "1-line `if`"
- An expression

```
a ? b : c
```

- means "if `a` is true use `b`. Otherwise use `c`"
    - In swak different cells/faces can use either `b` or `c`
    - because `a` is not necessarily homogeneous

replayTransientBC

- Writing `groovyBC` is a bit like programming
  - Sometimes mistakes happen
    - Not good if this happens at the end of a long run
- To test such boundary conditions there is `replayTransientBC`
  - Loads specified boundary conditions
  - Increments the time-step without solving anything
    - Updates the boundary conditions
  - Writes the field at the regular intervalls
- This allows checking whether the boundary condition works as expected
  - In a fraction of the time of the `real` solution
  - Works for non-swak4Foam boundary conditions as well

Preparing and running

From now on we don't mention the two steps:

1. `pyFoamPrepareCase.py`
   - optionally with `--no-mesh` if mesh creation is unnecessary
2. `pyFoamRunner.py`

We do this and get different plots

- And also different snapshots

Bigger area means higher temperature

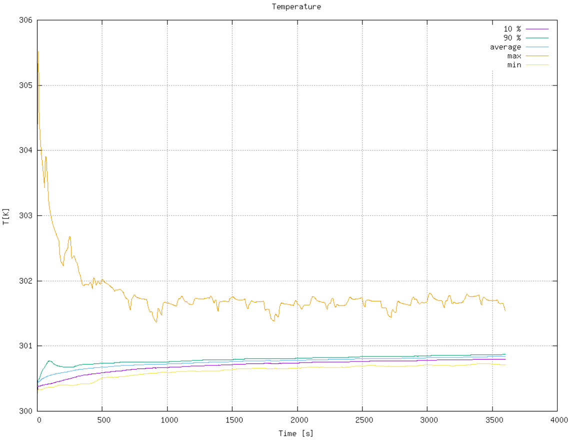

Figure: Temperature curves with the round/moving heater

Different wall temperatures

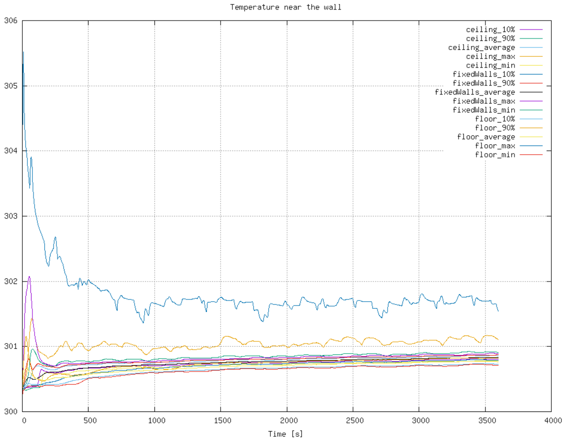

Figure: Wall temperatures change as well

Moving heater starting

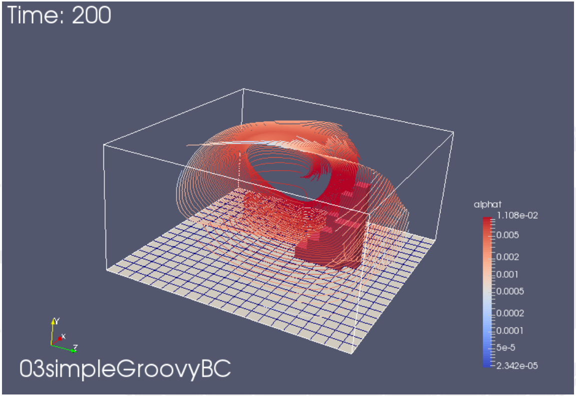

Figure: The moving heater in the beginning

Moving heater evolving

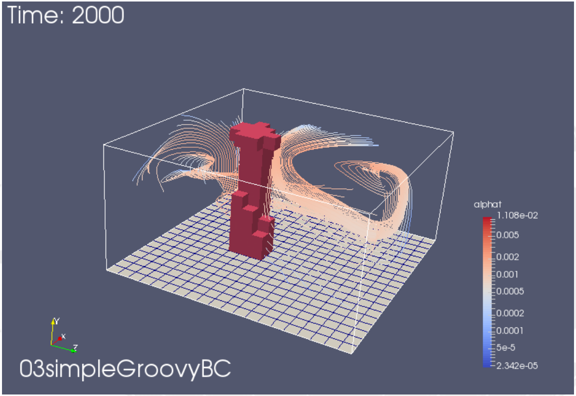

Figure: The moving heater moved on

Outline

Pro-tip: which fields are available?

- Function objects can only work with fields that are in memory
  - To get a list of those swak4Foam has a function object

functions in system/controlDict

```
whatIsThere {
    type listRegisteredObjects;
}
```

Output

```
Content of object registry region0
                   Name                          Type  Autowrite
======================= ============================= =========
                      K                      IOobject No
                    K_0                      IOobject No
          MRFProperties                      IOobject No
                      T                volScalarField Yes
<snip>
           thermo:alpha                      IOobject No
              thermo:mu                      IOobject No
             thermo:psi                      IOobject No
             thermo:rho                      IOobject No
thermophysicalProperties                   dictionary No
    turbulenceProperties                   dictionary No
```

Pro-tip: The *Banana* trick

If we don't know which function objects are there: we use the banana trick

functions in system/controlDict

```
gettingFunctionObjects {
    type banana;
}
```

Output

```
--> FOAM FATAL ERROR:
Unknown function type banana

Valid functions are :

90
(
abort
addForeignMeshes
addGlobalVariable
calculateGlobalVariables
clearExpressionField
coded
correctThermo
createSampledSet
createSampledSurface
dumpSwakExpression
....
```

New file for the heat flux

- We create a new file whose only purpose is the boundary condition
  - Calculates the heat flux on the wall

0.org/heatFlux

```
dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    ".*"
    {
        type groovyBC;
        value uniform 0;
        valueExpression "kappa*snGrad(T)";
        variables (
            "cpGas=1000;" // from thermoPhysicalProperties
            "kappa=cpGas*alphat;"
        );
        aliases {
            alpha thermo:alpha;
        }
    }
}
```

The aliases

- Some field names are incompatible with `swak4Foam`-expressions
  - Because of characters that are used for operators
  - Here it is the `:` in `thermo:alpha`
- In such cases specify a `aliases`-dictionary
  - the key is a "valid" name
  - the value is what you really want
- swak4Foam will use the "real" field when you specify the "alias" field

Calculate the flux

- This function object
  - loads the specified fields at startup
  - updates the boundary conditions at every time-step
  - writes the fields at write-times

controlDict

```
calculateCurrentFlux {
    type readAndUpdateFields;
    fields (
        heatFlux
    );
}
```

Calculating fluxes

Checking the fluxes (weighted sums)

```
> pyFoamPrepareCase.py . --no-mesh
...
> pyFoamRunner.py --clear --progress --auto  auto
...
> fieldReport -latestTime heatFlux -latestTime -noDoField  -doBoundary
<snip>
Time = 3600

 Reading Field heatFlux of type volScalarField

Patch field: floor
Size | Weight Sum                400 |          100
Range (min-max)            -0.242591 |      28.2346
Average | weighted           0.12034 |      0.12034
Sum | weighted               48.1362 |       12.034
Median | weighted          -0.132456 |    -0.132456

Patch field: ceiling
Size | Weight Sum                400 |          100
Range (min-max)            -0.916542 |    -0.255066
Average | weighted          -0.50377 |     -0.50377
Sum | weighted              -201.508 |      -50.377
Median | weighted             -0.492 |       -0.492

Patch field: fixedWalls
Size | Weight Sum                800 |          200
Range (min-max)                    0 |            0
Average | weighted                 0 |            0
Sum | weighted                     0 |            0
Median | weighted              5e-16 |        5e-16

End
```

Doing calculations after the fact

- `fieldReport` has two disadvantages
  1. it gives too much information
     - *sum* gives no sense for the temperature (not even weighted)
  2. it gives too little information
     - doesn't even calculate simple things like absolute magnitude of the velocity
- `funkyDoCalc` can do these things
  - Does calculations specified in a dictionary
  - Uses the data on disk
  - Can calculate on different things
    - internal fields, patches, cell sets, face sets etc
    - specified by `valueType` and additional parameters
  - Data can be written to a file as well
- we want to use this to calculate the heat fluxes
  - split be amount going out and in on each patch

Two heat fluxes for each patch

funkyDoCalc.heatFluxDir

```
floorIn {
    valueType patch;
    patchName floor;
    expression "heatFlux>0 ? heatFlux : 0";
    accumulations (
        integrate
    );
}
floorOut {
    $floorIn;
    expression "heatFlux<0 ? -heatFlux : 0";
}
ceilingIn {
    $floorIn;
    patchName ceiling;
}
ceilingOut {
    $floorOut;
    patchName ceiling;
}
fixedWallsIn {
    $floorIn;
    patchName fixedWalls;
}
fixedWallsOut {
    $floorOut;
    patchName fixedWalls;
}
```

Doing the calculation

Running the evaluation

```
> funkyDoCalc -time 0: system/funkyDoCalc.heatFluxDir -writeCsv
<snip>
Time = 3600
floorIn :  integrate=25.0994
floorOut :  integrate=13.0644
ceilingIn :  integrate=0
ceilingOut :  integrate=50.3787
fixedWallsIn :  integrate=0
fixedWallsOut :  integrate=0

6 CSV files written
End
> cat funkyDoCalc.heatFluxDir_data/floorIn.csv
Time,integrate
0,0
200,170.063
400,137.043
600,102.015
800,66.3082
1000,52.5273
<snip>
3200,36.583
3400,38.0648
3600,25.0994
```

Evaluating the heat fluxes

- get the fluxes on the walls
  - integrate is basically "weighted sum"
- Check whether "what goes in must go out"

system/controlDict

```
heatFluxes {
    $wallTemperatures;
    expression "heatFlux";
    accumulations (
        integrate
    );
}
totalHeatFlux {
    type swakExpression;
    valueType patch;
    patchName ceiling;
    verbose true;
    accumulations (
        average
    );
    expression "sum(area()*heatFlux)+heatFluxFloor";
    variables (
        "heatFluxFloor{floor}=sum(area()*heatFlux);"
    );
}
```

Remote variables

- If there is a {} between the variable name and the = then it is a remote variable
  - "Don't evaluate the expression here. Evaluate it elsewhere"
    - But store the value *here*
- If there is only a name between the {} it is a patch
  - In our case the `floor`
- Remote variables must be a single value (homogeneous)
  - Otherwise we'd have interpolation problems
- For details see *General variable specification* in the *Incomplete reference guide*
- My main application for this (but not here):
  - Calculate pressure drop between inlet and outlet

Plotting the heat-flux data

- a slave plot doesn't have its own plot window but plots into the window of the master
- alternateAxis specifies values that are on a different scale (on the right of the plot window)

customRegexp

```
heatFluxWall {
    theTitle "Heat␣flux";
    type dynamic;
    expr "Expression␣heatFluxes␣on␣(.+):␣␣integrate=(.+)";
    idNr 1;
    titles (
        sum
    );
    alternateAxis (
        total
    );
}
totalHeatFlux {
    type slave;
    master heatFluxWall;
    expr "Expression␣totalHeatFlux␣:␣␣average=(.+)";
    titles (
        total
    );
}
```

The heat fluxes

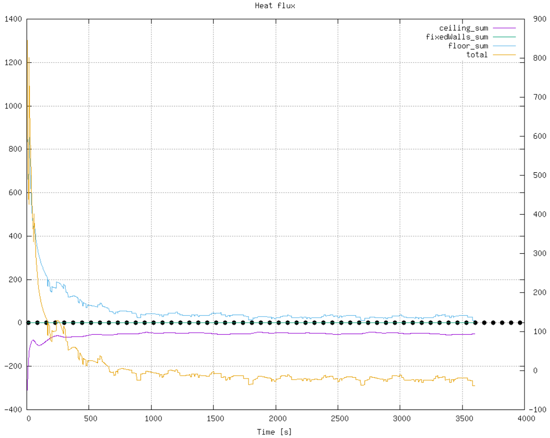

Figure: Heat fluxes over time

New state with heat fluxes

We use the `Warp by Scalar`-filter on `floor` and `ceiling` with the `heatFlux` field

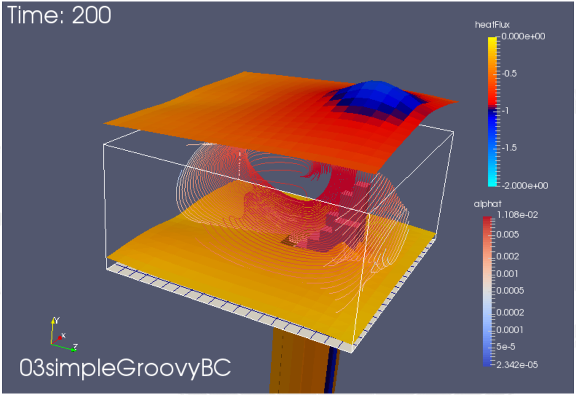

Figure: Heat flux at floor and ceiling

New state with heat fluxes . . . later

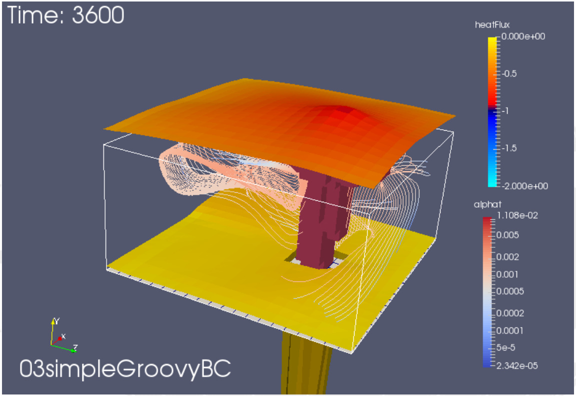

Figure: Heat flux at floor and ceiling in the end

This case

All we've done so far can be found in `03simpleGroovyBC.tar.gz` in the material

Outline

Outline

Discretization problems

- Sometimes the `swak`-expressions are "correct"
    - But the results are not
        - Because expressions are continuous but our calculations are discreet
- Here we show an example that is due to the rather coarse cells
    - Faces on the `floor` switch from 300 to 600
        - No intermediate values
- and a way to improve it

Getting the wall temperature

- To see the problem we add a plot of the patch values
  - But we don't use the `min` and `max` values
    - Because the $600K$ would have "destroyed" the plot

customRegexp

```
wallTemperatures {
    theTitle "Temperature␣on␣the␣wall";
    type dynamic;
    idNr 1;
    expr "Expression␣wallTemperatures␣on␣(.+):␣␣min=.+␣weightedQuantile0.1=(.+)␣<brk>
      <cont> weightedAverage=(.+)␣weightedQuantile0.9=(.+)␣max=.+";
    titles (
        "10%"
        average
        "90%"
    );
}
```

Wall temperature plot

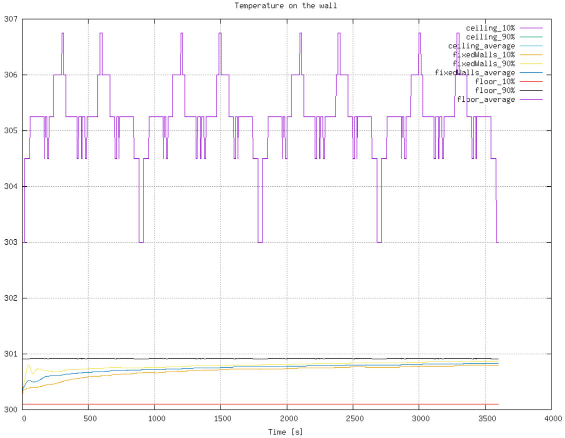

Figure: Jumps in the temperature on the wall

Smoothing by interpolation

- We calculate the same condition on the points
  - Interpolation with `toFace` gives better value for faces that are not fully "inside"
- Mixed with the `old` factor
  - Experimenting with the weighting could improve things further

0.org/T

```
floor
{
    type              groovyBC;
    value             uniform 300;
    variables (
        "center=vector(5,0,5);"
        "radiusFire=0.75;"
        "radiusCircle=1.5;"
        "radiant=2*pi*time()/3600;"
        "middle=center+radiusCircle*vector(sin(radiant),0,cos(radiant));"
        "tHigh=600;"
        "tLow=300;"
        "factor=mag(pos()-middle)<radiusFire ? 1 : 0;"
        "factorF=toFace(mag(pts()-toPoint(middle))<toPoint(radiusFire) ? toPoint(1) : <brk>
              <cont> toPoint(0));"
        // "factor=factorF;"
        "factor=0.5*(factorF+factor);"
    );
    valueExpression "tHigh*factor+tLow*(1-factor)";
}
```

Wall temperature plot smoothed

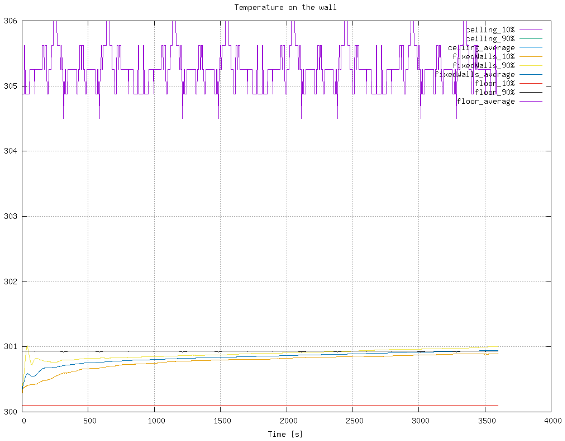

Figure: Jumps are much smaller (approximately a third)

Outline

Lumped boundary condition in OF+ v1612+

- In the ESI-version the same case has a `lumpedMass` boundary condition on the `ceiling`
  - Basically: "Ceiling is 1 ton with c_p 4100 and heated up by the room"
- In the Foundation-release this boundary condition does not exist
  - Here we try to implement it with `groovyBC`
- This example is the main reason why the Foundation fork was used in this training

0.orig/T in tutorial case

```
ceiling
{
    type            lumpedMassWallTemperature;
    kappaMethod     fluidThermo;
    kappa           none;
    mass            1000;
    Cp              4100;
    value           uniform 300;
}
```

Stored variables

- Regular entries in `variables` forget their values between time-steps
- When we specify them in the `storedVariables`-list they don't
  - They are even saved and read on restart
    - So our lumped-condition is restartable
- Specification of a stored variable needs two things

  name

  intialValue the value that should be used when the variable has never been set before
- When the variable is on the right of a = the stored value is used
- The last value the variable is set to is stored for the next time-step
- `storedVariables` are aware that there can be multiple iterations per time-step
  - old values are from the last time. Not the last iteration

Re-implementation with groovyBC

- Calculating the total heat flux and updating the temperature of the ceiling accordingly

0.org/T

```
ceiling
{
    type groovyBC;
    valueExpression "TLump";

    variables (
        "mass=1000;"
        "cpSolid=4100;"
        "cpGas=1000;" // from thermoPhysicalProperties
        "kappa=cpGas*alphat;"
        "Q=sum(area()*kappa*snGrad(T));"
        "TLump=TLump-deltaT()*Q/(mass*cpSolid);"
    );
    storedVariables (
        {
            name TLump;
            initialValue "300";
        }
    );
    aliases {
        alpha thermo:alpha;
    }
    value           uniform 300;
}
```

Ceiling heats up (slightly)

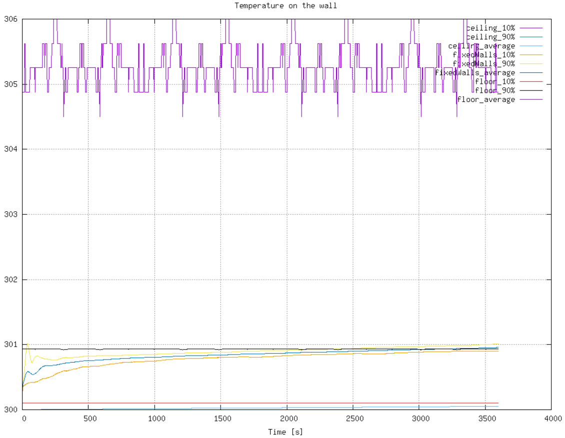

Figure: Ceiling temperature on the bottom

Heat fluxes

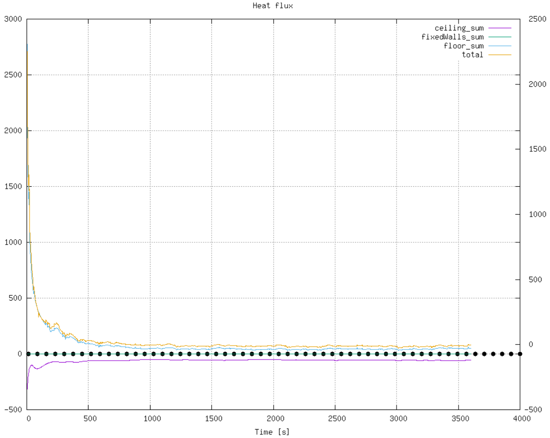

Figure: Heat fluxes differ slightly

Outline

Low temperature outside our room

- Instead of an adiabatic wall we want something more realistic
  - 7 degrees C outside. A cool day
  - Not perfectly isolated walls (ceofficient `h`)
- But windows are usually less insulated than walls
  - One solution: create separate patches
    - This is much work
  - What we do: variation of `h`
- But first lets run the window-less case

0.org/T

```
fixedWalls
{
    type externalWallHeatFluxTemperature;
    value uniform 300;
    Ta uniform 280;
    h  uniform 0.01;
    kappaMethod fluidThermo;
}
```

Room temperature doesn't rise that much anymore

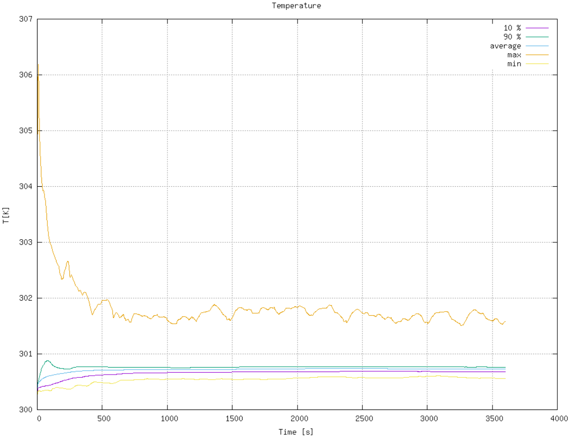

Figure: Room temperature almost constant

Wall temperatures below room

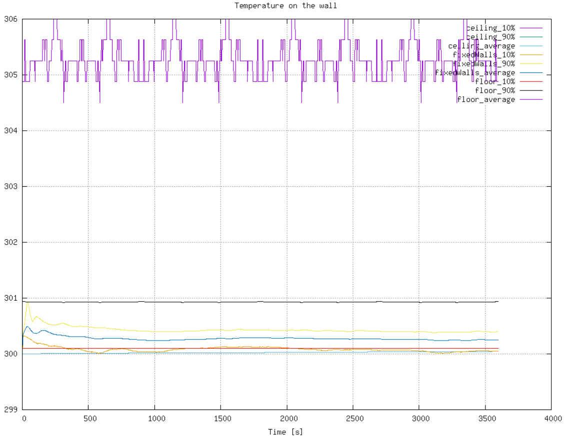

Figure: On the `fixedWall` the temperature falls

Heat flux were none was before

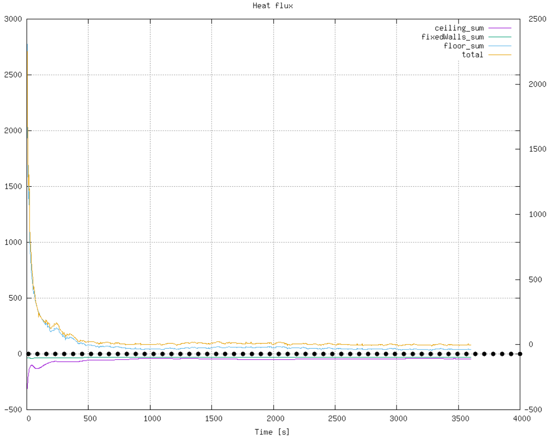

Figure: On the `fixedWall` there is now non-zero flux

Directed (outside) flux for post-processing

system/controlDict

```
calculateCurrentFlux {
    type readAndUpdateFields;
    fields (
        heatFlux
        heatFluxDirected
    );
}
```

0.org/heatFluxDirected

```
dimensions      [0 0 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    ".*"
    {
        type groovyBC;
        value uniform (0 0 0);
        valueExpression "normal()*kappa*snGrad(T)";
        variables (
            "cpGas=1000;" // from thermoPhysicalProperties
            "kappa=cpGas*alphat;"
        );
        aliases {
            alpha thermo:alpha;
        }
    }
}
```

Heat flux with constant coefficient

New state with `Warp by Vector`-filter in Paraview

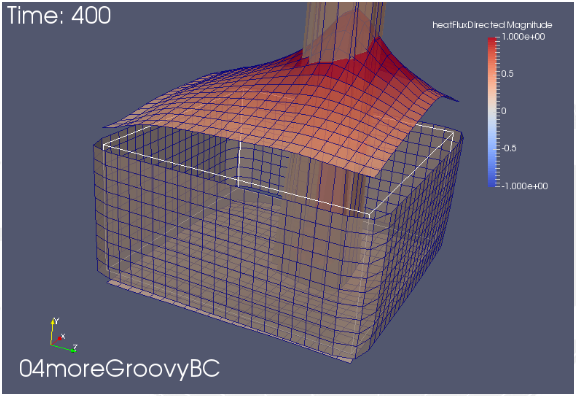

Figure: Surface outside wall: heat goes out

Specifying the coefficient

- funkySetBoundaryField is like funkySetFields
  - but for boundaries
  - can set other things that value
    - specify with target
- The dictionary structure is quite similar

system/funkySetBoundaryFieldDict.setWall

```
transferCoeff {
    field T;
    expressions
    (
        {
            target h;
            patchName fixedWalls;
            variables (
                "minY=1;"
                "maxY=2.5;"
                "r=mag(vector(pos().x,0,pos().z)-vector(5,0,5));"
                "thres=mag(vector(5,0,2.5));"
            );
            expression "(pos().y<maxY && pos().y>minY && r<thres) ? 0.1 : 0.01";
        }
    );
}
```

Preparing

- We re-add a preparation script
- `writeBoundarySubfields` is utility to create separate fields from boundary condition specifications
  - Here we say "Write `h` and `Ta` into fields so that we can post-process them"
  - Nice to debug boundary conditions

caseSetup.sh

```
#! /bin/sh

funkySetBoundaryField -time 0 -dict funkySetBoundaryFieldDict.setWall
writeBoundarySubfields -time 0 -subfields "h:scalar,Ta:scalar" T
```

Heat transfer coefficient

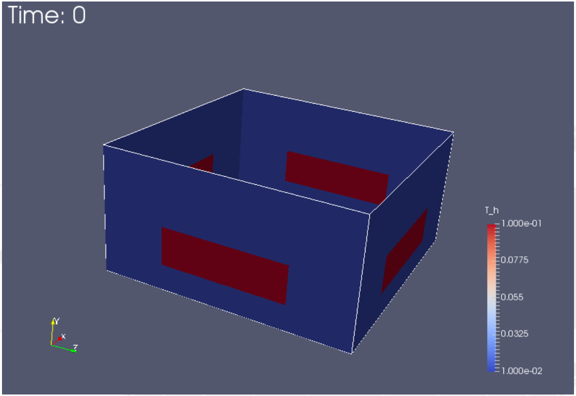

Figure: Our expression seems to have worked: high heat transfer on "windows"

Heat flux with lower coefficients

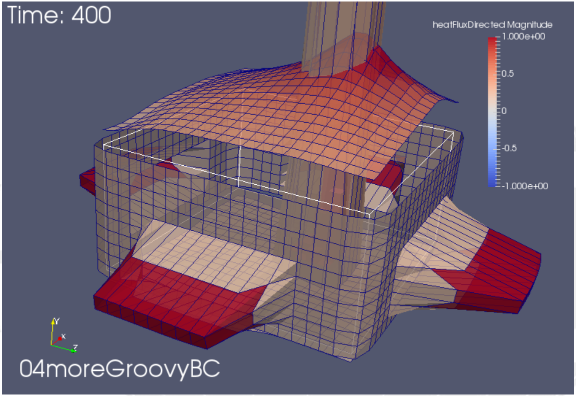

Figure: The "windows" have a big effect

Windows are bad

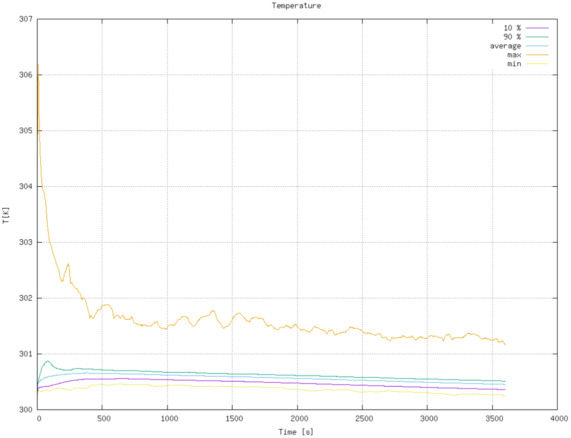

Figure: The bad isolation makes the room temperature drop

Windows are really bad

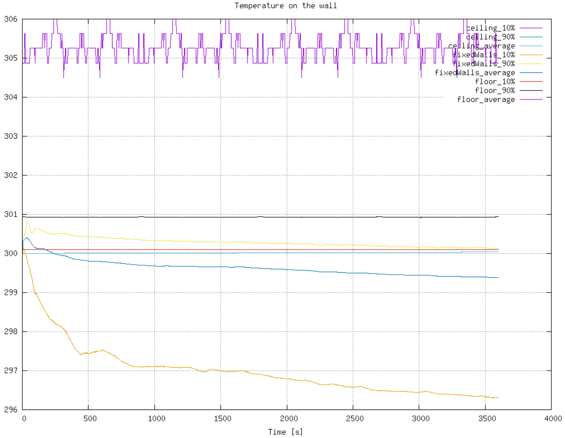

Figure: Wall temperatures drop even more

Total flux is wrong

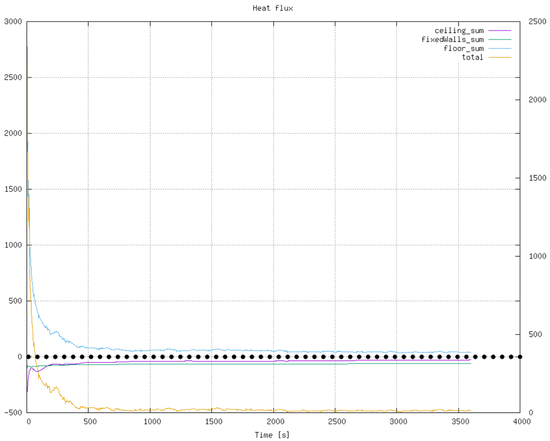

Figure: What is the reason for this?

The --long-progress

- Newer PyFoam-version have the option `--long-curses`
  - Prints all the numbers that `PyFoam` usually gets in one summary
  - Formatted in a "table-like" fashion for easier reading
- Especially nice with `--curses`
  - Instead of rapidly scrolling text you get a table that updates itself
  - Possibility to scroll in it with the list is too long
  - An additional line in the header which lines of the output are shown
- This option is for you with you think the numbers are more interesting than the graphs

A long and complicated curse

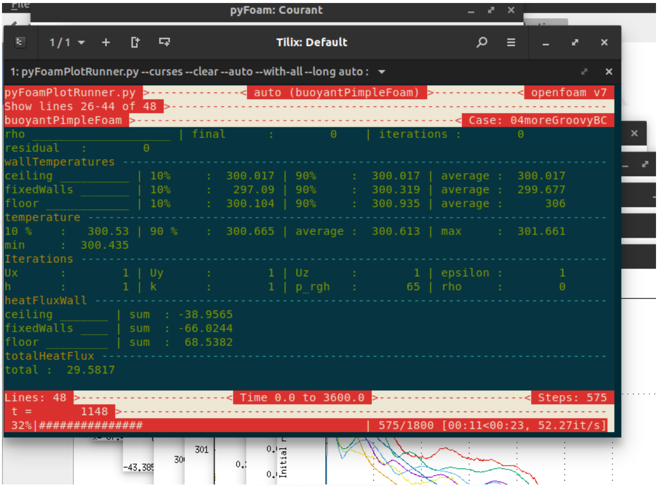

Figure: Output with `--long-progress` and `--curses`

This case

All we've done so far can be found in `04moreGroovyBC.tgz` in the material

Outline

There is more

- We've seen only parts of PyFoam and swak4Foam
- Other available things are
  - In swak4Foam
    - Adding lagrangian particles by function objects
    - Arbitrary source terms
    - Control of the solution
    - Execute Python programs
    - ...
  - PyFoam
    - Support for parallel runs
    - Flexibly preparing cases
    - Controlling runs over the net
    - Rewriting dictionaries
    - ...

Further reading

- This presentation only covered parts of PyFoam and swak4Foam, but there is further information available:
  - On the OpenFOAM-wiki:
    - https://openfoamwiki.net/index.php/Contrib/swak4Foam in the section *Further Information* are links to previous presentations
    - https://openfoamwiki.net/index.php/Contrib/PyFoam in section *Other material*
  - The Examples directory of the swak-sources
  - Did I mention the *Incomplete reference guide* for swak?
  - The --help-option of the PyFoam-utilities

Further presentations

- `pyFoamPrepareCase.py` can handle lots of things
  - With something called *templates*
  - See "Automatic case setup with `pyFoamPrepareCase`" from the Ann Arbor Workshop 2015
    - an updated version was given at the Shanghai Workshop 2018
- We skipped the parts about writing data
  - These are explained in another presentation
    - "PyFoam for the lazy" from Guiamares Workshop in 2016
- Cool things can be done with swak4Foam to change parameters during the run
  - See "State and solution" from the Exeter Workshop 2017
- There will be another presentation on advanced `swak4Foam` topics today

Goodbye to you

Thanks for listening

Questions?

License of this presentation

Authors of this document are:

Bernhard F.W. Gschaider original author and responsible for the strange English grammar. Contact him for a copy of the sources if you want to extend/improve/use this presentation