swak4Foam and PyFoam for solver developers

Make your life easier - the physics is hard enough

Bernhard F.W. Gschaider

HFD Research GesmbH

UCD Dublin, Ireland
9. June 2021

Outline I

1. Introduction
   - This presentation
   - Who is this?
   - What are we working with
   - Before we start
   - What we simulate
2. Setup
   - OpenFOAMs developer support
   - PyFoams helps
3. Prototyping
   - Solving simple PDEs
   - Other
4. Development
   - The solver

Outline II

- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Outline

1. Introduction
   - This presentation
   - Who is this?
   - What are we working with
   - Before we start
   - What we simulate
2. Setup
   - OpenFOAMs developer support
   - PyFoams helps
3. Prototyping
   - Solving simple PDEs
   - Other
4. Development
   - The solver
   - What happened
   - Irregular conditions
   - fvOptions
5. Optimizing
   - High-level profiling
   - Pyfoam support for profiling output
6. Conclusion

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

The topic

- This presentation shows how swak4Foam and PyFoam can help you
  - with your OpenFOAM-development
  - finding problems in your cases
- Highlights
  - Post-mortem dumpy for failed cases
  - Finding out what uses most time

Intended audience and aim

- Advanced OpenFOAM-users
  - people who develop their own OpenFOAM-solvers
    - I won't explain the C++-stuff in detail
  - know a little about swak4Foam and PyFoam
    - yesterdays basic training is sufficient
- But these is interesting information for non-developers as well

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Bernhard F.W. 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

    ansibleFoamInstallation "Universal build script for OpenFOAM"
  - 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
  - ...

Who is Ignaz

- Earlier presentations on swak4Foam and PyFoam were following Ignaz Gartengschirrl
  - A CFD engineer
  - One of the first and most enthusiastic users of swak4foam and PyFoam
  - We may call him by first name
- Some time ago Ignaz stopped his appearances in the presentations
  - Official explanation: changed career (Pogo dance instructor and head-banging trainer)
- Now Ignaz is back
  - The few concerts due to COVID were not good for his business
- Ignaz is now working for a company *Warm Rooms Ltd*
  - They want him to develop an new combustion model
    - It is un-physical but it is patented

Warning: no realistic physics

- Realistic models have a problem: They are either
  - already implemented in OpenFOAM
  - rather complicated
- So the "combustion" model in this presentation is
  - simple
  - doesn't resemble any real physical phenomena
    - outside of comic-books and over-CGIed movies
- So don't
  - use it
  - reference it

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

What is PyFoam

- PyFoam is a library for
  - Manipulating OpenFOAM-cases
  - Controlling OpenFOAM-runs
- It is written in Python
- Has very few dependencies
  - Doesn't even need an OpenFOAM-installation
- 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/index.php/Contrib/PyFoam

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

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Command line examples

- In the following presentation Ignaz 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 in 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
```

Getting onto the same page

- We need a machine with
  - OpenFOAM 2012
    - but older versions work as well
    - and other forks like `foam-extend` or `OF 8` (but some of the sources might have to be adapted)
  - swak4foam
  - PyFoam
  - Text editors: `emacs`, `vim`, `gedit`

Open a shell and set us up for work

Assuming that you have a machine with those things installed

```
> mkdir swakAndPyFoam
> cd swakAndPyFoam
> . ~/OpenFOAM/OpenFOAM-v2012/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 an image prepared for this training
  - Found at
    `https://hub.docker.com/r/bgschaid/openfoam_by_ansible`
  - Based on Ubuntu 18.04 LTS
  - OpenFOAM v2012
  - Most recent release (`2021.06`) of PyFoam
  - Most recent release (`2021.05`) of swak4Foam
  - has no ParaView. Sorry
- The image was prepared with `https://openfoamwiki.net/index.php/Installation/Ansible`

Pulling the Docker-Image

Problems here:

- The image is over 2 Gig.
  - Depending on your network this might take some time
- You have to have `docker` installed on your machine

Pulling the container

This will download the container the first time around

```
> docker pull bgschaid/openfoam_by_ansible:training_swak_pyfoam_4devs_ofw16
```

Getting the script

```
> wget https://bit.ly/ofw16docker -O runFoamContainer.sh
> chmod a+x runFoamContainer.sh
```

The actual URL for the script is `http://hg.code.sf.net/p/openfoam-extend/ansibleFoamInstallation/raw-file/226d8688cbaa/scripts/runFoamContainer.sh`

Starting the container

```
> ./runFoamContainer.sh bgschaid/openfoam_by_ansible:training_swak_pyfoam_4devs_ofw16
```

After that you're on a shell inside the container

What `runFoamContainer.sh` does

The purpose of this script is to make using the Docker container as painless as possible

- Without an argument the script lists the locally available containers compatible with the script
- With an image name it starts the image in a new container
- mounts the working directory on the host machine to `/foamdata` on the container
  - data written to that directory is written to the host machine
    - and can be read during the next start of the machine
- Sets the user id of the user in the container to the id of the user on the host machine
  - Can read and write the same files as the host user

Starting the container

This demonstrates how data written inside the container is written to the host machine (`rechenknecht001` is the name of the host. `testuser` is the name of the user on the host)

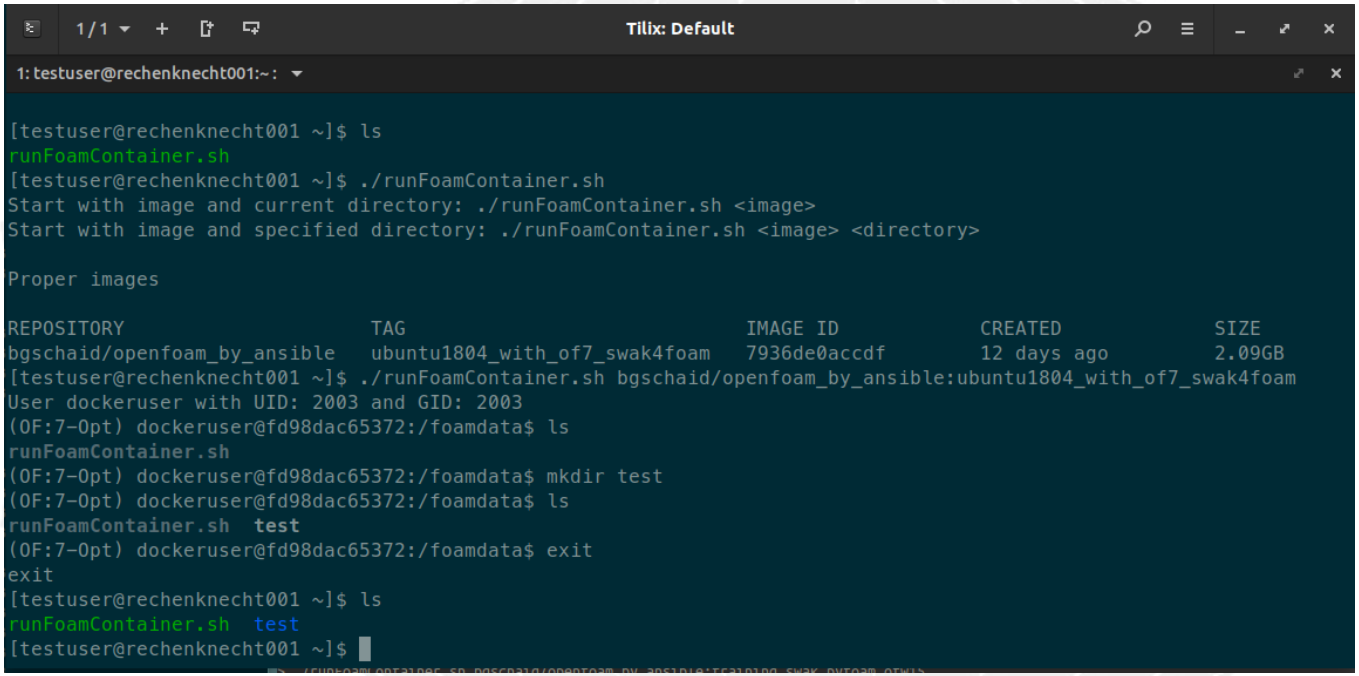


Figure: Docker container started and data written to local machine (version numbers differ)

Fix: Compiling on the container

Currently the `dockeruser` is not allowed to create a directory in the container. To fix this

- Create the directory

```
sudo mkdir /home/dockeruser/OpenFOAM/dockeruser-v2012
```

- Make `dockeruser` the owner

```
sudo chown dockeruser:dockeruser /home/dockeruser/OpenFOAM/docke
```

Getting the Material

With docker

The docker image has the 4 stages in the directory `/Examples`

```
> runFoamContainer.sh bgschaid/openfoam_by_ansible:training_swak_pyfoam_4devs_ofw16
User dockeruser with UID: 2000 and GID: 2000
> (OF:v2012-Opt) dockeruser@188b002cec4a:/foamdata$ ls /Examples
01prototypeTimeTrigger
02rhoTimeTriggerFoam
03timeTriggeredPitzDaily
04rhoTimeTriggerFoamProfile
```

Non-docker

The main stages of the presentation are archived in a tar

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

```
> wget
> wget http://bit.ly/swakPy4Devs16 -O PyFoamSwak4Devs_Dublin2021_Material.tar.gz
> tar xvzf PyFoamSwak_Dublin2021_Material.tar.gz
01prototypeTimeTrigger.tar.gz
02rhoTimeTriggerFoam.tar.gz
03timeTriggeredPitzDaily.tar.gz
04rhoTimeTriggerFoamProfile.tar.gz
```

Alternate URL: `https://openfoamwiki.net/images/d/db/PyFoamSwak4Devs_Dublin2021_Material.tar.gz`

Make sure PyFoam is working

First Ignaz wants to be sure he has the newest and shiniest version of PyFoam

- 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: Linux | b79d027672c3 | 5.4.0-73-generic | #82~18.04.1-Ubuntu SMP Fri Apr 16 15:10:02 UTC 2021 | x86_64<brk>
      <cont> | x86_64

Python version: 3.6.9 (default, Jan 26 2021, 15:33:00)
[GCC 8.4.0]

Python executable: /usr/bin/python3

Python 3 is supported with PyFoam
PYTHONPATH is not set

Location of this utility: /usr/local/bin/pyFoamVersion.py

Version 2012 (reported as number 2012 )Fork openfoamplus of the installed 1 versions:
openfoam-v2012 : /home/openfoam/OpenFOAM/OpenFOAM-v2012

pyFoam-Version: 2021.6

Path where PyFoam was found (PyFoam.__path__) is ['/usr/local/lib/python3.6/dist-packages/PyFoam']

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

Installed libraries:
cython                        :  No     Not used. Maybe will by used later to spped up parts of PyFoam
cProfile                      :  Yes
docutils                      :  No     Not necessary. Needed for RestructuredText to HTML conversion
Gnuplot                       :  No     Not a problem. Version from ThirdParty is used
....
```

Make sure swak4Foam is installed

Now Ignaz wants to see whether swak4Foam is working

- Calls 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     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  8
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
Build  : 8
Exec   : funkySetFields
Date   : Jun 01 2021
Time   : 17:32:28
Host   : "b79d027672c3"
PID    : 221
I/O    : uncollated
Case   : //foamdata
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: 2021.05 (Release date: 2021-05-31)
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


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


    From function main()
    in file funkySetFields.C at line 713.

FOAM exiting
```

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Trigger warning: Comic book physics

- The next slides describe stuff that doesn't exist in the real world
- Go to another training if
  - the sight of fantasy-equations disturbs you
  - you are under 18
- Thanks

Ignaz needs a new combustion model

- *Warm Rooms Ltd* are in the business of keeping rooms warm
- They want to develop a new "heating fluid"
  - Starts heating some time after being injected into the room
    - Until a maximum rate is reached
  - High temperatures "regenerate" the fluid (it gets younger)
- Nobody knows how to produce such a fluid
  - But Ignaz is asked to simulate it

The model equations

- Source term added to the energy equation

$$\rho E_h R$$

- Reaction rate

$$R = \begin{cases} 0, & \tau < \tau_{start} \\ R_{max} \frac{\tau - \tau_{start}}{\tau_{end} - \tau_{start}}, & \tau_{start} \leq \tau \leq \tau_{end} \\ R_{max}, & \tau > \tau_{end} \end{cases}$$

- Fluid age (flow time)

$$\frac{\partial \rho \tau}{\partial t} + \nabla(\rho \tau \vec{v}) = 1 - (\rho \lambda T_{extra} \tau)$$

- With the excess temperature

$$T_{extra} = \begin{cases} 0, & T < T_{thres} \\ T - T_{thres}, & T_{thres} \leq T \end{cases}$$

Outline

1. Introduction
   - This presentation
   - Who is this?
   - What are we working with
   - Before we start
   - What we simulate
2. Setup
   - OpenFOAMs developer support
   - PyFoams helps
3. Prototyping
   - Solving simple PDEs
   - Other
4. Development
   - The solver
   - What happened
   - Irregular conditions
   - fvOptions
5. Optimizing
   - High-level profiling
   - Pyfoam support for profiling output
6. Conclusion

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

By developers for developers

- OpenFOAM is still very much developer-oriented
  - Nice for developers
  - Not so nice for people who don't want to program
- Has some very useful things built in
  - Some of them are switched on

The importance of having a Debug version

- The environment variable `WM_COMPILE_OPTION` selects the way OpenFOAM is compiled
  - Opt The usual option.
    - Fast
  - Debug The option for developers
    - Terribly slow
- What er the advantages of slow
  - Additional debug symbols in the code
  - Almost every `[]` operation is range-checked
    - Run fails if the index is outside the array
- Have a `Debug`-version when you're developing
  - Even if your program seems to be bug-free compile and run it at least once in `Debug`
    - Sometimes crazy bug fall out

The FOAM_ABORT environment variable

- This makes every error an abort
  - You get a stack-trace
- When you're developing you don't want the solver to be tolerant

FOAM_SIGFPE crashes on bad math

- This environment variable makes sure that *Floating point exceptions* are raised and properly
  - Some compilers generate code that calculates on even if the results are `NaN` (Not a number)
    - Once you have these you won't get proper results
    - For instance: division by zero
- Make sure that this variable is set to `true`

FOAM_SETNAN: the brown M&M

- This environment variable helps finding uninitialized memory
  - Together with `FOAM_SIGFPE`
- When set the memory allocation for fields does some additional work
  - It fills the allocated memory with `NaN` values
- If that memory is not initialized but read and used for calculation a *Floating Point Exception* is raised
  - If there was no `NaN` there would have been a random number
- If the memory was initialized everything is fine

This is similar to the famous "Van Halen insist on huge bowls of M&M with all the brown ones removed" story. Google it if you don't know it

Stack-traces

- These are printed when the program aborts
- Prints the function calls that lead to the problem
  - The "lowest" 4 can be ignored (error handling code)
- With `WM_COMPILE_OPTION=Debug` it prints source files and line numbers

A missing parameter

```
FOAM aborting (FOAM_ABORT set)

#0  Foam::error::printStack(Foam::Ostream&) at /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OSspecific/POSIX/<brk>
      <cont>printStack/printStack.C:237
#1  Foam::IOerror::exitOrAbort(int, bool) at /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/lnInclude/IOerror.C<brk>
      <cont>:205
#2  Foam::IOerror::exit(int) at /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/lnInclude/IOerror.C:252
#3  Foam::Ostream& Foam::operator<< <Foam::IOerror, int>(Foam::Ostream&, Foam::errorManipArg<Foam::IOerror, int>) at<brk>
      <cont> /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/lnInclude/errorManip.H:125
#4  bool Foam::dictionary::readEntry<double>(Foam::word const&, double&, Foam::keyType::option, bool) const at /home<brk>
      <cont>/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/lnInclude/dictionaryTemplates.C:?
#5  Foam::Time::readDict() at /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/db/Time/TimeIO.C:263
#6  Foam::Time::setControls() at /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/db/Time/Time.C:192
#7  Foam::Time::Time(Foam::word const&, Foam::argList const&, Foam::word const&, Foam::word const&, bool, bool) at /<brk>
      <cont>home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/db/Time/Time.C:565
#8  Foam::Time::Time(Foam::word const&, Foam::argList const&, bool, bool) at /home/openfoam/OpenFOAM/OpenFOAM-v2012/<brk>
      <cont>src/OpenFOAM/lnInclude/TimeI.H:38
#9  ? at /home/openfoam/OpenFOAM/OpenFOAM-v2012/src/OpenFOAM/lnInclude/createTime.H:3
#10  __libc_start_main in /lib/x86_64-linux-gnu/libc.so.6
#11  ? at ??:?
```

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Having different Foam-versions

There are reasons to have multiple OpenFOAM-versions installed

- Different forks have different capabilities
- To reproduce old results
- To check whether a problem also occurs with old versions
- Because a project is ported to a new version

PyFoam helps Ignaz to seamlessly work with these different version

Selecting versions

- Normally an OpenFOAM-version is "activated" on the shell like this

. ~/OpenFOAM/OpenFOAM-v2012/etc/bashrc

- This is "permanent"

pyFoam allows setting a Version just for the next command

```
> pyFoamRunner.py --foamVersion=8 checkMesh
 Reading regular expressions from /slowdata/LinkInHome/Projects/LaTexDocs/Vortraege/Dublin2021/swakPyFoam4Developers<brk>
       <cont>/exampleCaseSolver/timeTriggeredPitzDaily/customRegexp
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  8
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
Build  : 8
Exec   : checkMesh
Date   : Jun 09 2021
```

Unknown version

```
> pyFoamRunner.py --foamVersion=of8 checkMesh
<snip>
PyFoam.Error.FatalErrorPyFoamException: FatalError in PyFoam: 'PyFoam FATAL ERROR on line 185 of file /home/bgschaid<brk>
      <cont>/Projects/PyFoam/src/PyFoam/FoamInformation.py: Can't find basedir for OpenFOAM-version of8 in extend<brk>
      <cont>-4.1, openfoam-7, openfoamplus-v1912, openfoam-6, openfoam-8, openfoam-1.7.1, openfoam-5.0, openfoamplus-<brk>
      <cont>v1806, openfoam-1.5, openfoamplus-v2012, openfoamplus-v1812, openfoam-2.3.0, openfoam-1.3, openfoam<brk>
      <cont>-2.3.1, openfoamplus-plus, openfoam-dev, openfoam-1.1, openfoamplus-v2006, openfoamplus-v1906, extend<brk>
      <cont>-4.1-old'
```

Opt or Debug

- The `WM_COMPILE_OPTION` can be selected with the options `--force-opt` and `--force-debug`
- With these a `--foamVersion` always has to be specified
  - As a replacement `--currentFoamVersion` keeps the currently active version

Run a debug-version

```
> pyFoamRunner.py --currentFoamVersion --force-debug --clear simpleFoam
```

Execute a utility that does not exist in this version

```
> pyFoamRunner.py --foamVersion=2212 --force-opt superMeshFoam
```

pyFoamExecute.py for non-Foam commands

- Sometimes we want to run utilities that are not OpenFOAM-solvers

Compiling a debug-version

```
> pyFoamExecute.py --currentFoamVersion --force-debug wmake mySolver
```

The current version has no paraFoam

```
> pyFoamExecute.py --foamVersion=of7 paraFoam
```

Simple controlDict-modifications

- The Runner utilities have options that modify controlDict temporarily
  1. Utility modifies the controlDict
  2. Runs solver
  3. Reverts changes on controlDict
- Options for that are

  –write-all-timesteps write everything

  –run-until set a different endTime

Check the startup

```
> pyFoamRunner.py --run-until=0.001 --write-all-timesteps --clear auto
```

Outline

1. Introduction
   - This presentation
   - Who is this?
   - What are we working with
   - Before we start
   - What we simulate
2. Setup
   - OpenFOAMs developer support
   - PyFoams helps
3. Prototyping
   - Solving simple PDEs
   - Other
4. Development
   - The solver
   - What happened
   - Irregular conditions
   - fvOptions
5. Optimizing
   - High-level profiling
   - Pyfoam support for profiling output
6. Conclusion

Why prototyping?

- Prototyping: building a simpler version of the finished product (solver)
  - To demonstrate that it will work
  - To see where the problems will be
  - To test different approaches
- swak4Foam has a number of tools to prototype a solver

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Adding flow time

- Ignaz wants to add a "flow time" to an existing solver to see if that works

The "general" transport equation

The function object `solveTransportPDE` solves the transport equation

$$\frac{\partial \rho T}{\partial t} + \div(\phi, T) - \nabla \lambda \nabla T = S_{expl} + S_{impl} T$$

- The terms of the equation can be specified as expressions
  - rho `swak`-expression and dimension for ρ
  - lambda same for λ
  - source/sourceImplicit S_{expl} and S_{impl}
  - phi Name of the `scalarSurfaceField` that is ϕ (sorry. Currently no expression)

fieldName name of T

The "flow time" equation

Simpler version of the general transport equation

- "transported quantity" is a time
- every second one second is added to the time
- there is no diffusion

$$\frac{\partial \rho\tau}{\partial t} + \nabla(\rho\tau\vec{v}) = 1$$

The base case

Ignaz uses the `pitzDaily` case for `rhoPimpleFoam` because

- it is a simple, well-known case
- sufficiently non-trivial
- but also not too big

and adds the equation

Writing it as a function object

Ignaz adds the transport equation

In functions in controlDict

```
solveTime {
    type solveTransportPDE;
    libs (swakFunctionObjects);
    solveAt timestep;
    fieldName time;
    aliases {
        flowTime time;  // avoid clash with time() function
    }
    steady false;
    rho "rho" [1 -3 0 0 0 0 0];
    diffusion "0" [1 -1 -1 0 0 0 0];
    source "1" [1 -3 0 0 0 0 0];
    phi "phi" [1 0 -1 0 0 0 0];
    relaxUnsteady false;
}
```

The necessary field file

Necessary because `solveTransportPDE` needs it for the boundary conditions

0.org/time

```
dimensions      [0 0 1 0 0 0 0];

// internalField   uniform -0.1;
internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    ".+"
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

Options of the PDEs

It is possible to specify when and how the equation is solved

steady ignore the dime derivative and relax the equation

solveAt when to solve thew equation

- for instance: sometimes a steady flow only needs to be solved at the start

Result: Start temperature

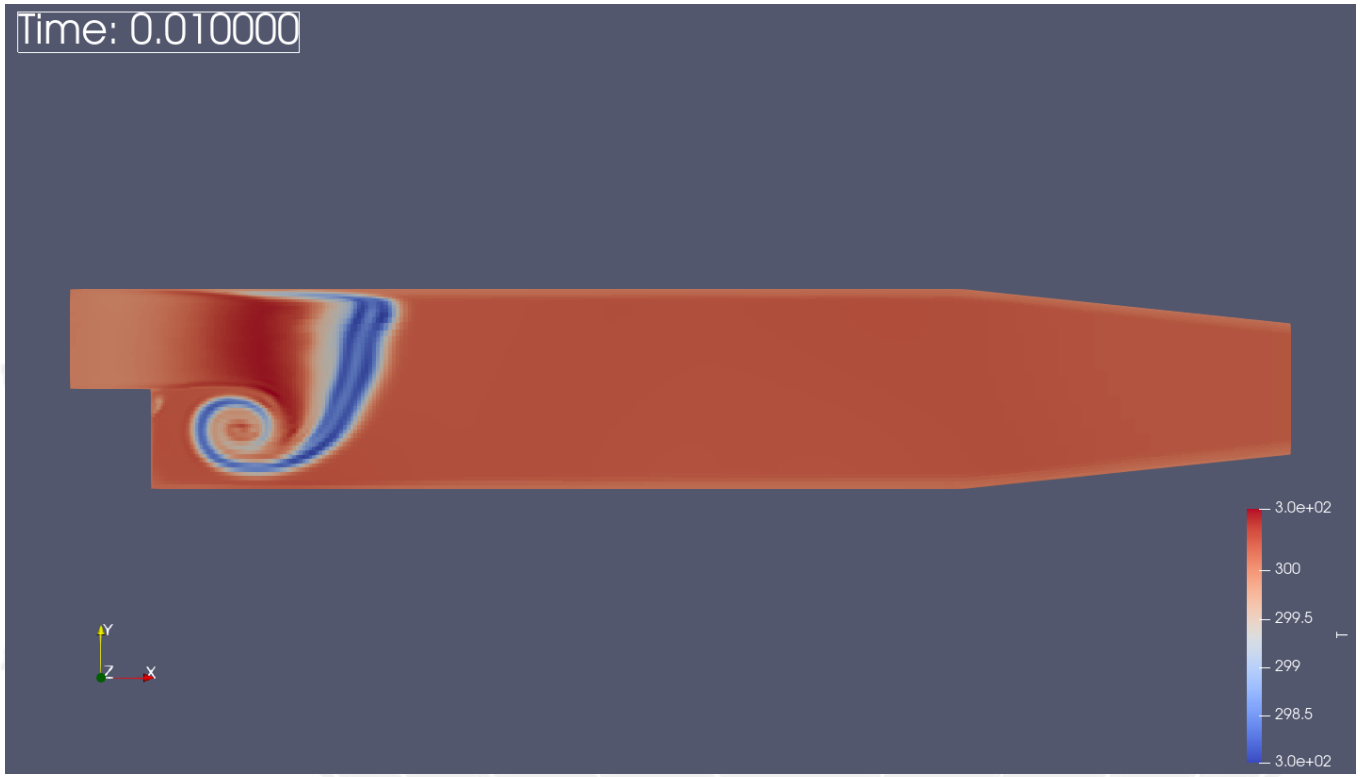


Figure: Temperature at an early time-step

Result: End temperature

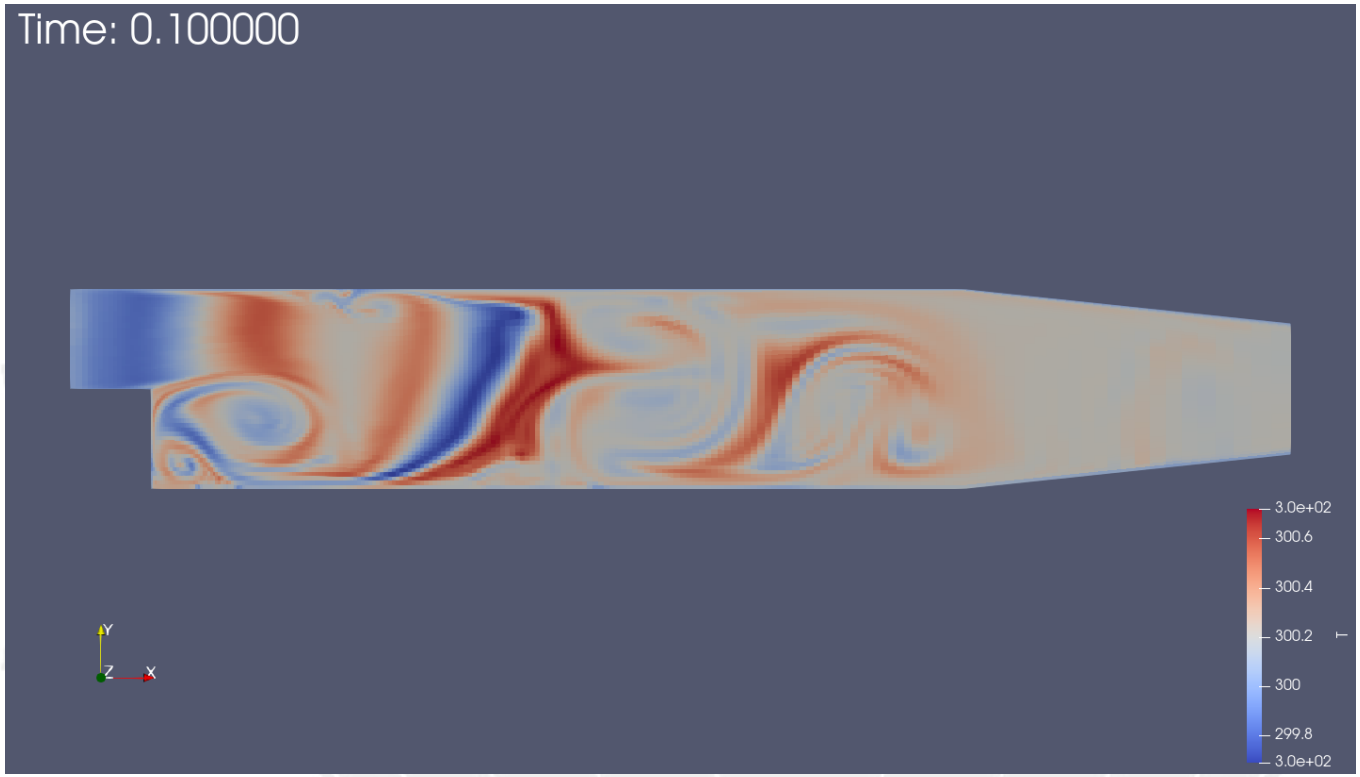


Figure: Temperature in the end

Graphs are helpful

- Often during development graphs are helpful to see that something is wrong
  - No need to open ParaView
- For that "ordinary" swak4Foam/PyFoam is sufficient
  - use a `swakExpression` function object
  - add a `customRegexp` for PyFoam to do the plotting
- But that is described in the introductory presentation

Result: flow time at the start

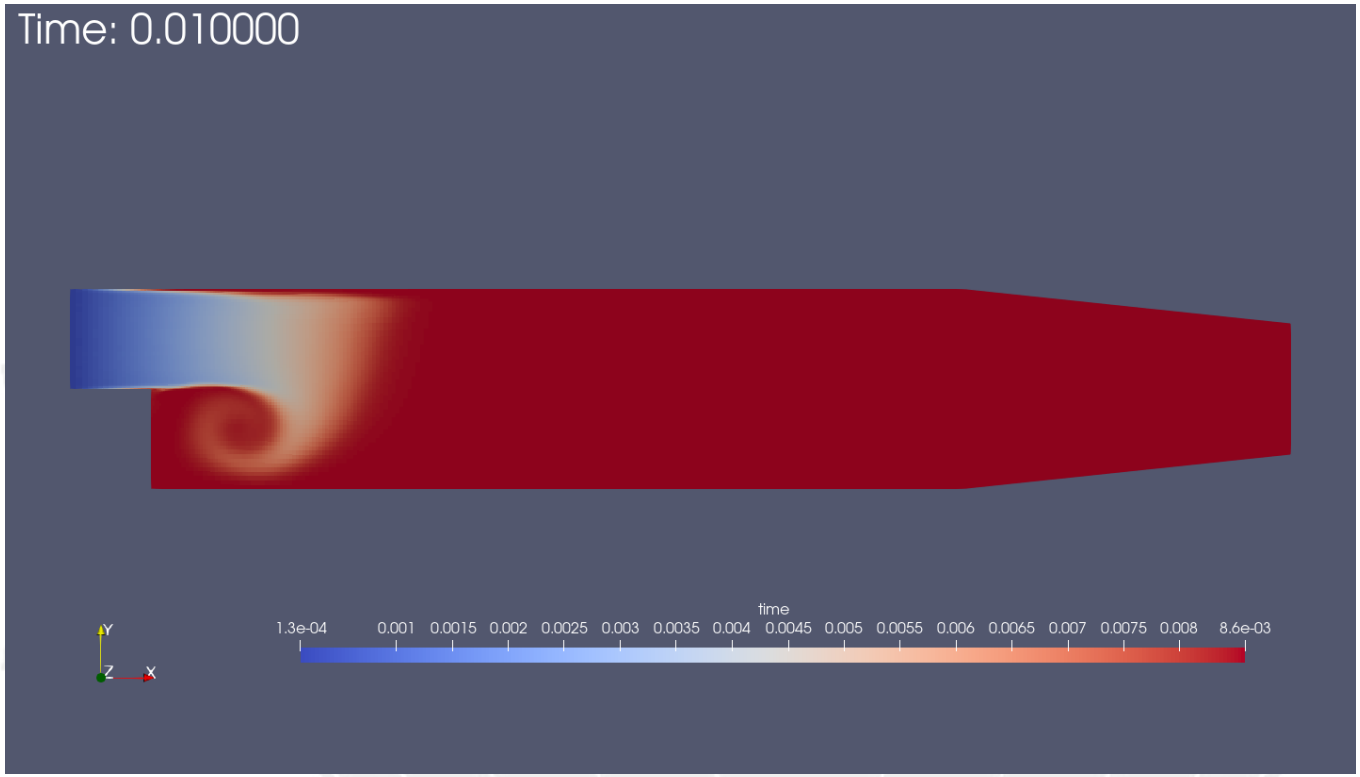


Figure: Flow time at the start

Result: flow time converged

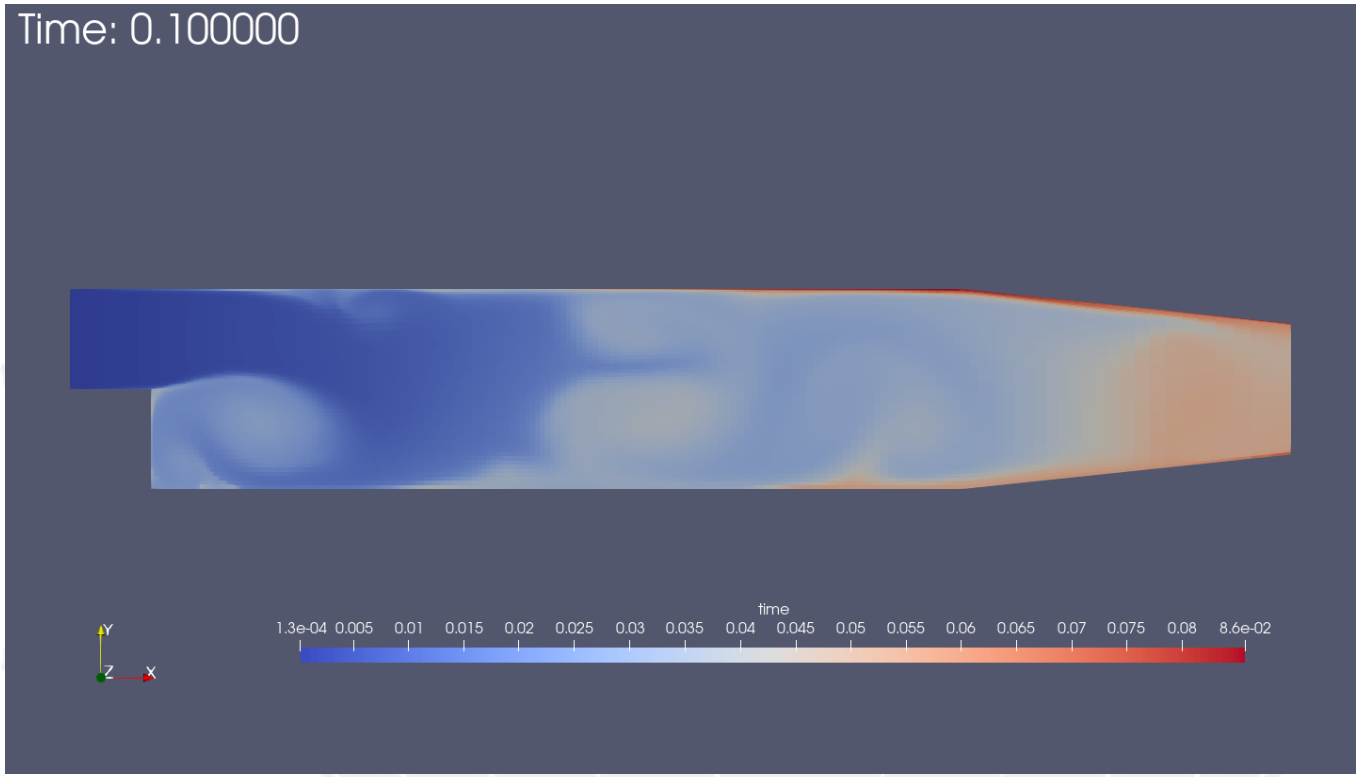


Figure: Flow time in the end

Result: temperature time graph

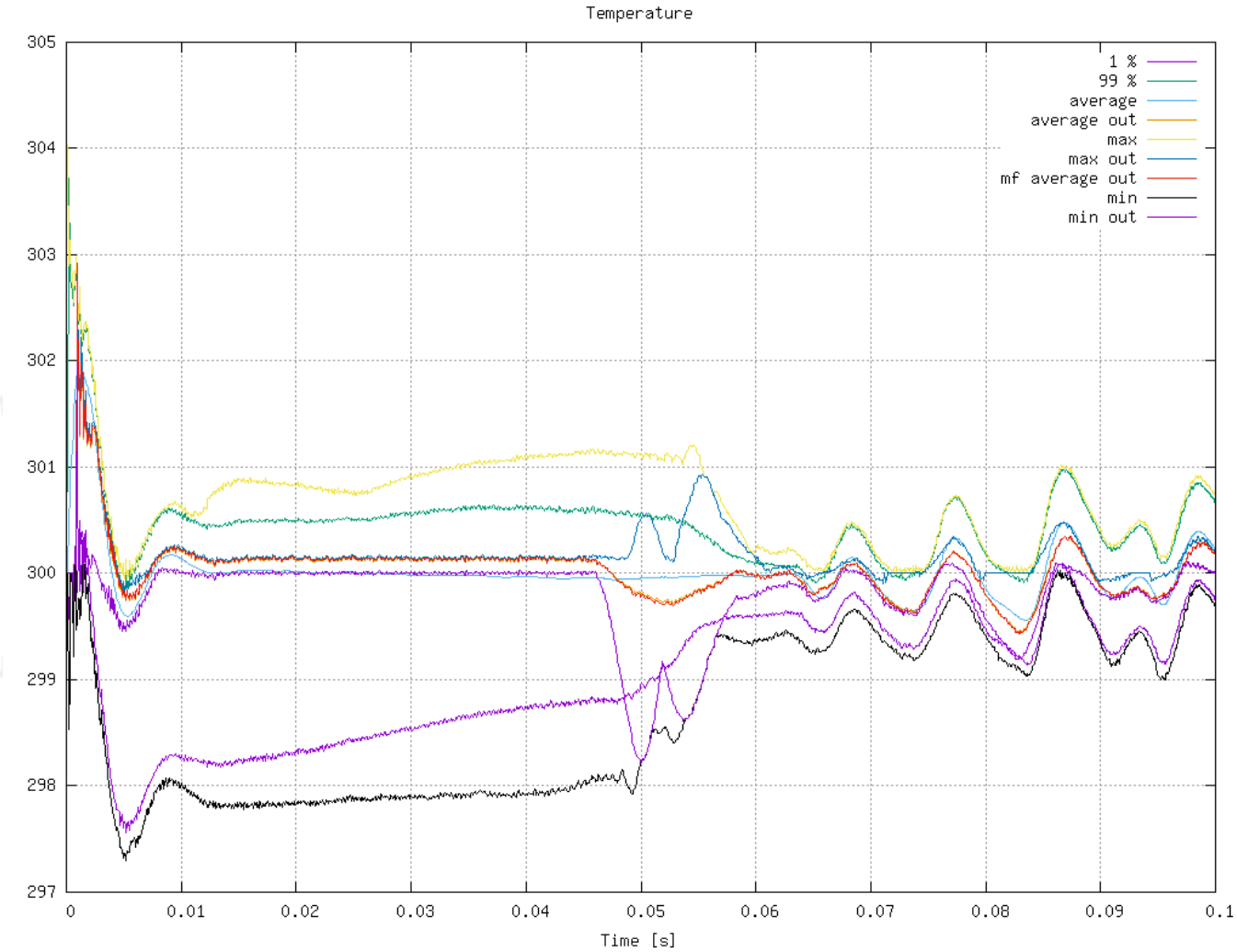


Figure: Evolution of the temperature

Result: flow time graph

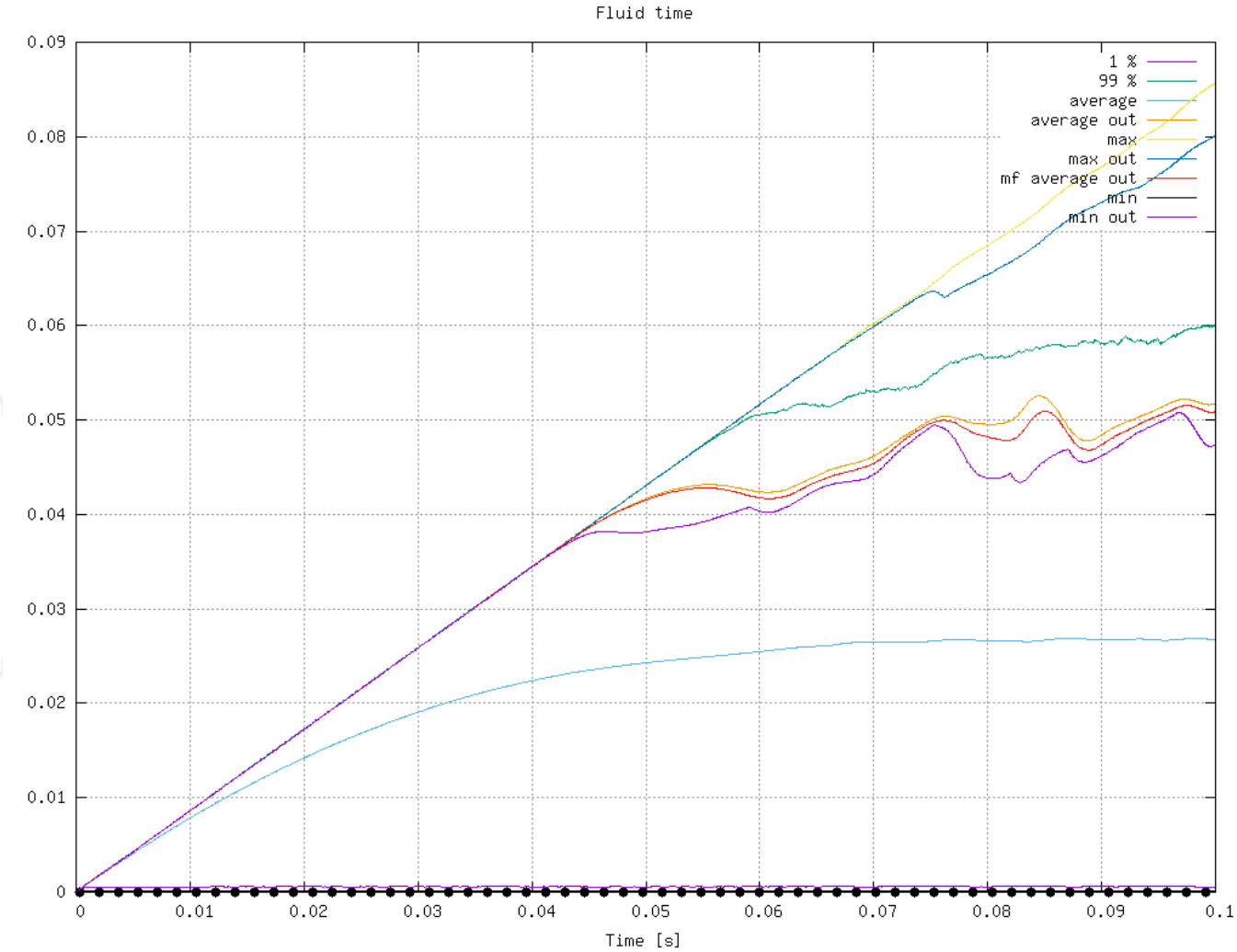


Figure: Evolution of the flow time

The high flow time

- Ignaz noticed that the flow time near the walls is very high
  - Very low velocities: fluid stays there for a long time
  - No diffusion: it is not "transported" out
- He decides to try a "limiting" approach
- Changes

```
source "1" [1 -3 0 0 0 0 0];
```

to

```
source "flowTime < 0.02 ? 1 : 0" [1 -3 0 0 0 0 0];
```

Result: modified flow time in the end

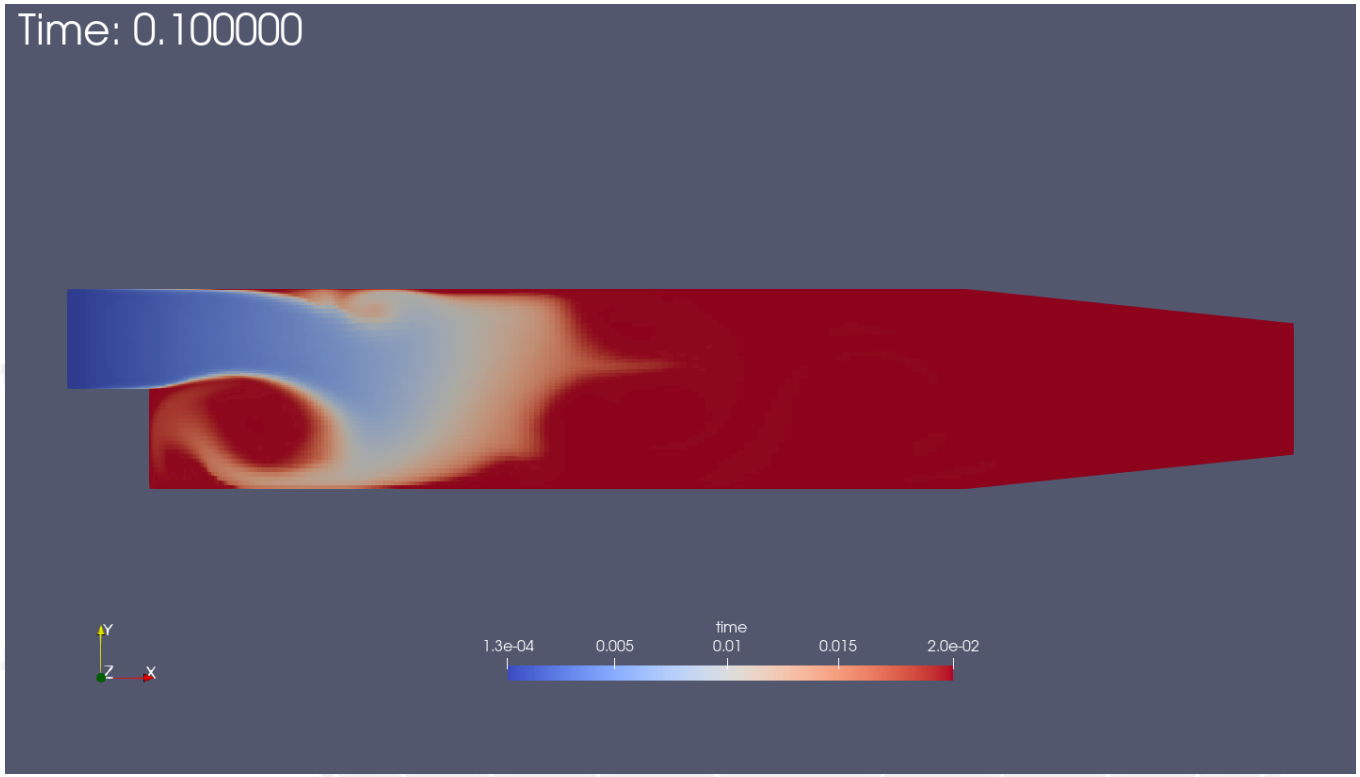


Figure: Flow time in the end

Result: modified flow time graph

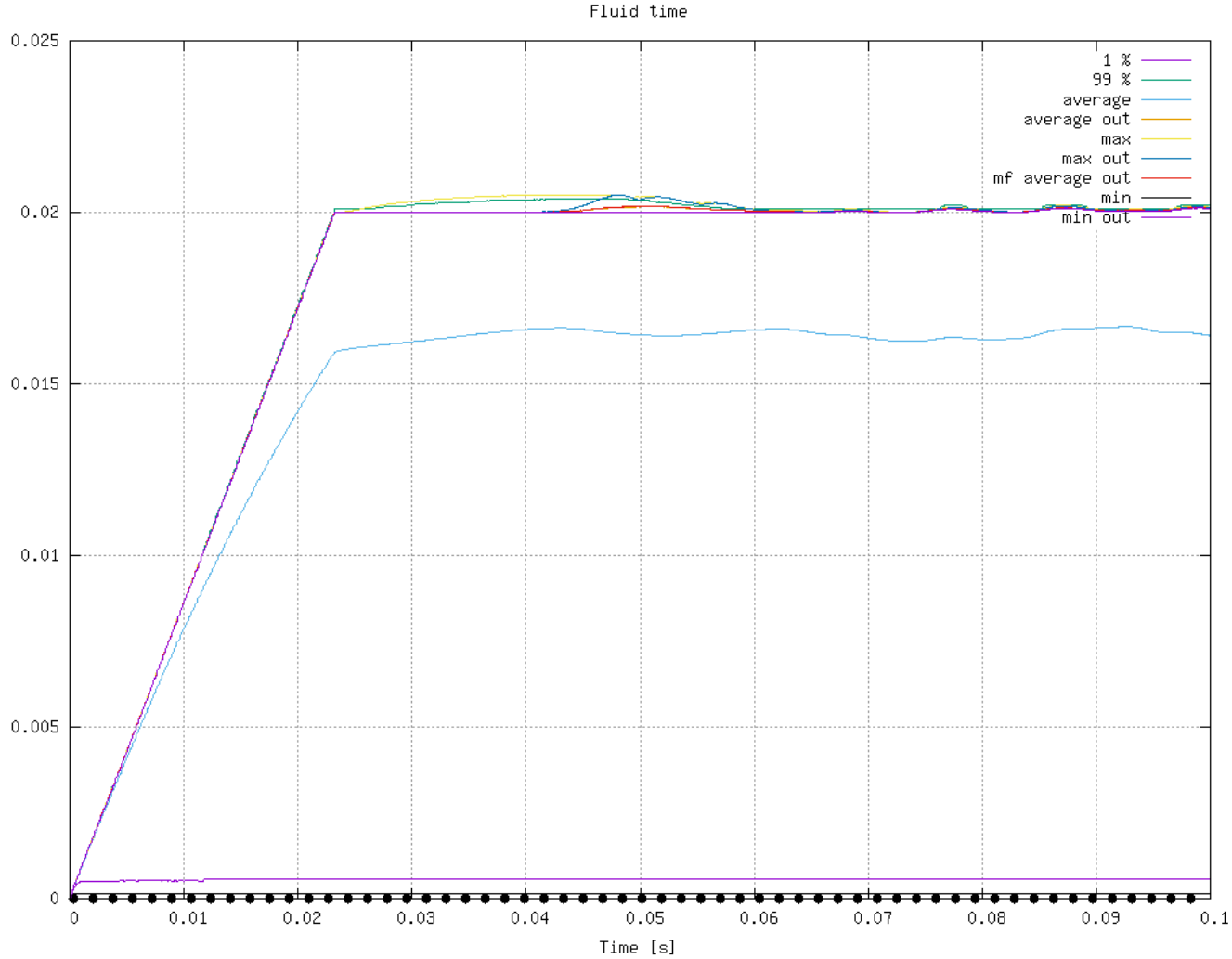


Figure: Evolution of the flow time

The "general" laplacian equation

- `solveLaplacianPDE` solves the equation

$$\frac{\partial \rho T}{\partial t} - \nabla \lambda \nabla T = S_{expl} + S_{impl} T$$

- Can be used for heat-conduction/diffusion problems

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Boundary conditions

- the most versatile tool to prototype boundary conditions is `groovyBC`
  - can do non-trivial things like
    - coupling with other parts of the model
    - storing of values
    - time-delay values

For an example see the Advanced presentation from the Duisburg-Workshop

Additional source terms

- swak4Foam has `fvOptions` where the source term can be an expression
- Of course it is necessary that the used solver supports fvOptions

The filter-case

A complete example for prototyping a model for

- porous filter
  - with changing properties
- particle flow

can be found in the "State and solution" presentation from the 2017 Workshop in Exceter

Example case

The case with the prototyped is in the file `01prototypeTimeTrigger.tar.gz` (or `/Example/01prototypeTimeTrigger` in the Docker image)

Outline

1. Introduction
   - This presentation
   - Who is this?
   - What are we working with
   - Before we start
   - What we simulate
2. Setup
   - OpenFOAMs developer support
   - PyFoams helps
3. Prototyping
   - Solving simple PDEs
   - Other
4. Development
   - The solver
   - What happened
   - Irregular conditions
   - fvOptions
5. Optimizing
   - High-level profiling
   - Pyfoam support for profiling output
6. Conclusion

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Basis: rhoPimpleFoam

Ignaz uses this solver because

- it is transient
- solves the energy equation
  - "only" a source term has to be added
  - ... and an equation for the flow time

Material

- The source code for the solver can be found in `02rhoTimeTriggerFoam.tar.gz`
- The example case in `03timeTriggeredPitzDaily.tar.gz`
- Or in the corresponding subdirectories of `/Examples` of the docker image

Added to the solver

Modifications to the .C file

Field construction and parameter reading

```
#include "createFields.H"
#include "createTimeFields.H" // added
#include "createFieldRefs.H"
#include "createRhoUfIfPresent.H"
#include "readParameters.H"    // added
```

Solve the flow time

```
#include "UEqn.H"

 #include "timeEqn.H"    // added

 #include "EEqn.H"
```

Reading parameters

`readParameters.H` reads the "physical" parameters

`reaction_start_time` when the reaction starts

`reaction_max_time` when the `max_rate` is reached

`max_rate` maximum reaction rate

`reaction_energy` energy set free by the reaction

`decyay_temperature` after which temperature the *flow time* decays

`decay_speed` how fast that happens

The time equation `timeEqn.H`

Calculating the extra temperature

Calculates the "high" temperature

```
volScalarField excess_temperature(
    "excess_temperature",
    T-decay_temperature
);

forAll(excess_temperature, cellI) {
    if(excess_temperature[cellI] < 0) {
        excess_temperature[cellI] = 0;
    }
}
```

Solving the equation

Similar to the prototyped model

- the second source-term is new

```
fvScalarMatrix timeEqn
(
    fvm::ddt(rho, time)
    + fvm::div(phi, time)
    ==
    fvOptions(rho, time)
    +
    rho*dimensionedScalar("increase", <brk>
        <cont>dimTime/dimTime, 1)
    -
    fvm::Sp(
        rho
        *
        decay_speed
        *
        excess_temperature,
        time)
);
```

Calculating the energy source

How fast will the reaction happen?

timeEqn.H

```
forAll(reactionRate, cellI) {
    if(time[cellI] < reaction_start_time) {
        reactionRate[cellI] = 0;
    } else if (time[cellI] > reaction_max_time) {
        reactionRate[cellI] = max_rate;
    } else {
        reactionRate[cellI] =
            max_rate*(time[cellI] - reaction_start_time)
            /
            (reaction_max_time - reaction_start_time);
    }
}
```

Modified energy equation

Only the last source term is new

One term added to `EEqn.H`

```
fvScalarMatrix EEqn
(
    fvm::ddt(rho, he) + fvm::div(phi, he)
  + fvc::ddt(rho, K) + fvc::div(phi, K)
  + (
        he.name() == "e"
      ? fvc::div
        (
            fvc::absolute(phi/fvc::interpolate(rho), U),
            p,
            "div(phiv,p)"
        )
      : -dpdt
    )
  - fvm::laplacian(turbulence->alphaEff(), he)
 ==
    fvOptions(rho, he)
    +
    reaction_energy * reactionRate * rho
);
```

The parameters

The actual values used

constant/timeTriggeredProperties

```
reaction_start_time  0.02;

reaction_max_time  0.04;

max_rate  1;

reaction_energy [0 2 -2 0 0 0 0] 1e8;

decay_temperature [0 0 0 1 0 0 0] 400;

decay_speed [0 0 -1 -1 0 0 0] 1;
```

Result: flow time

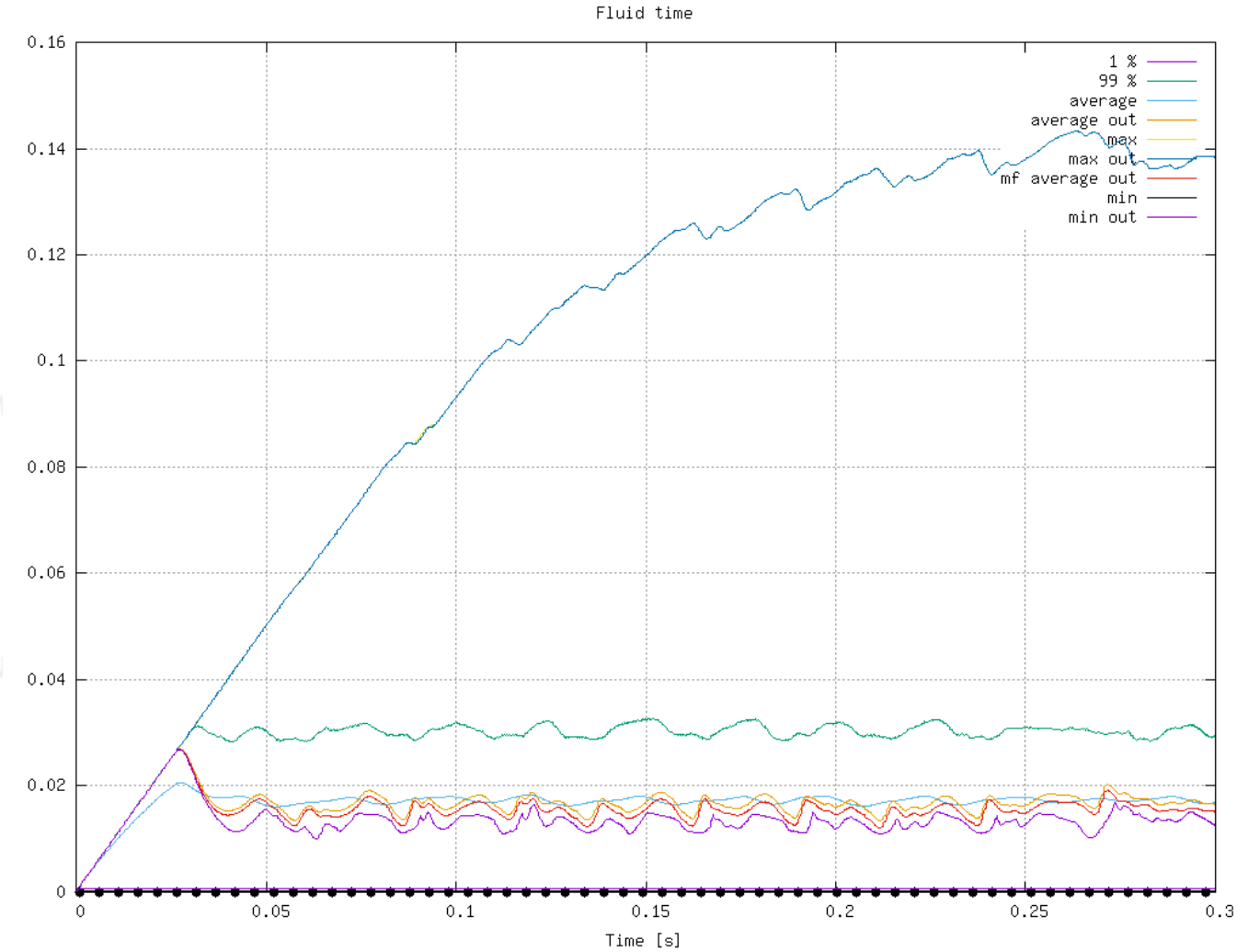


Figure: Flow time during the run

Result: temperature

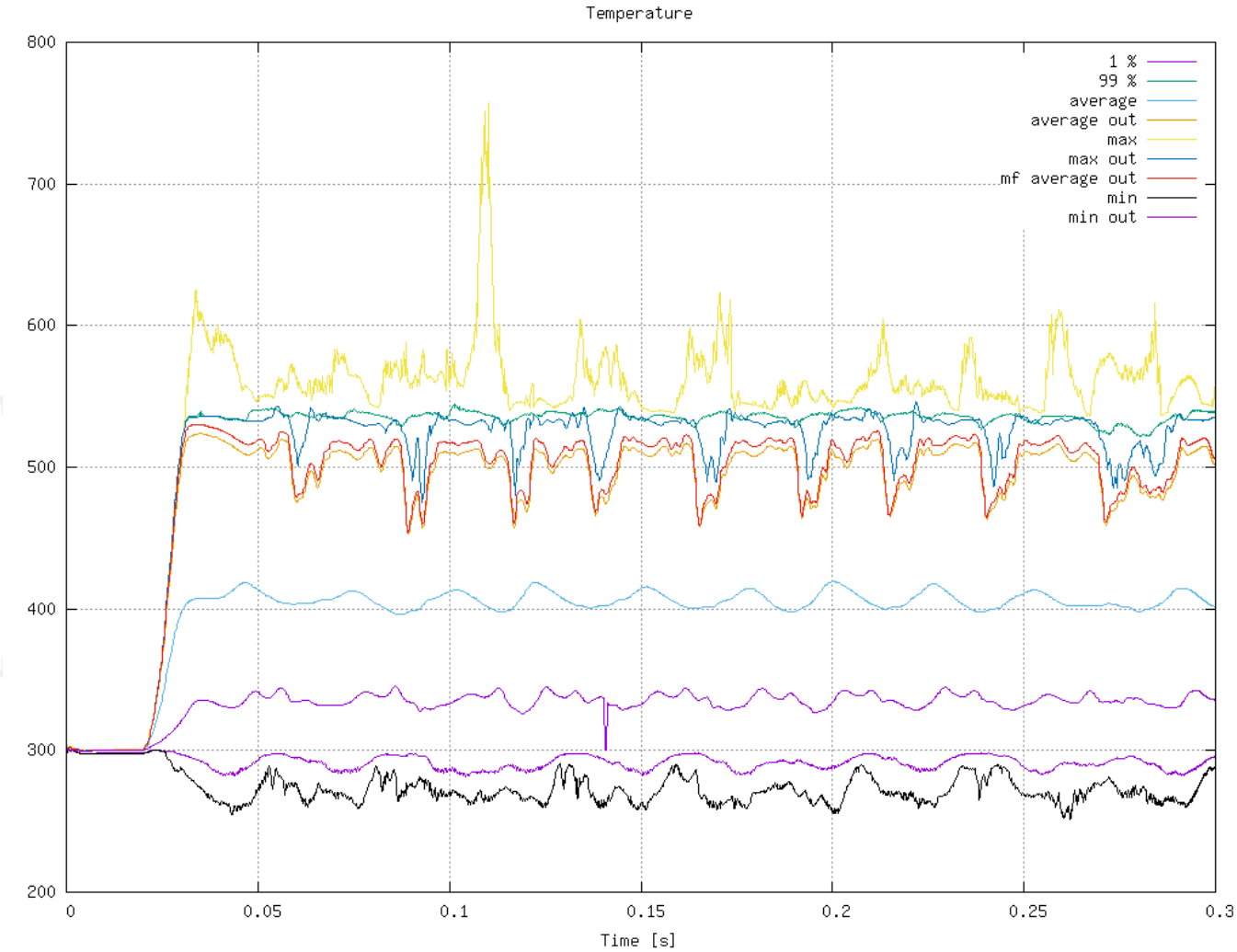


Figure: Temperature during the run

Result: reaction rate

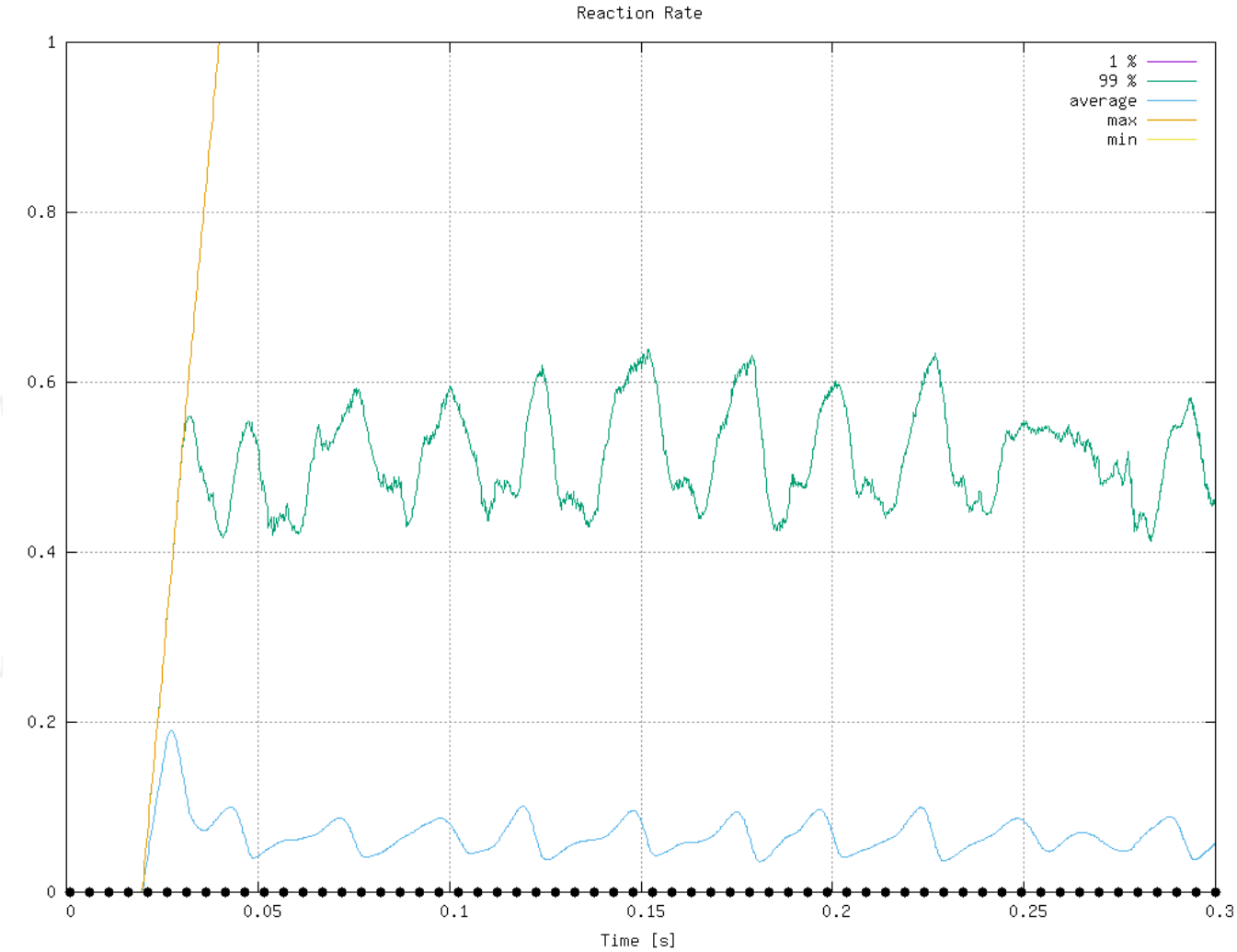


Figure: Reaction rate during the run

Result: "converged" state temperature

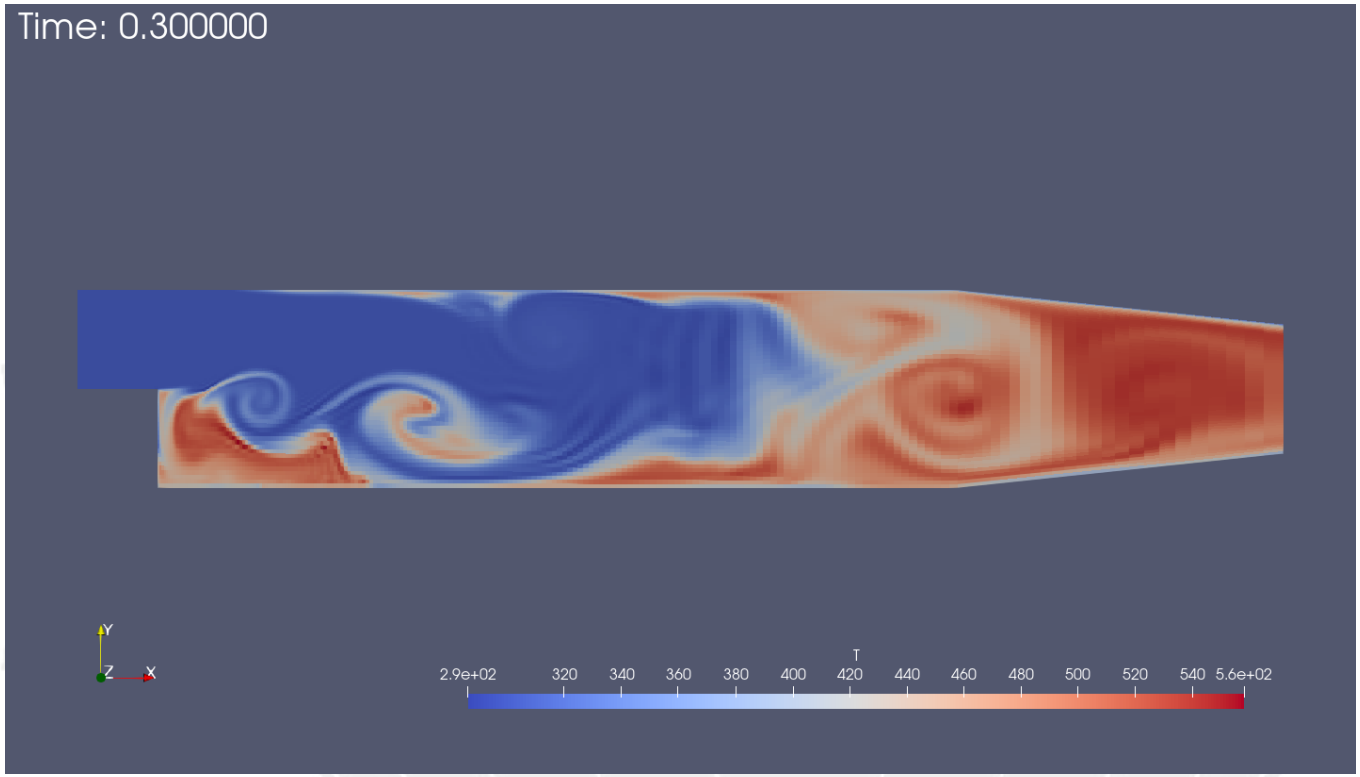


Figure: Temperature at the end of the simulation

Result: "converged" state flow time

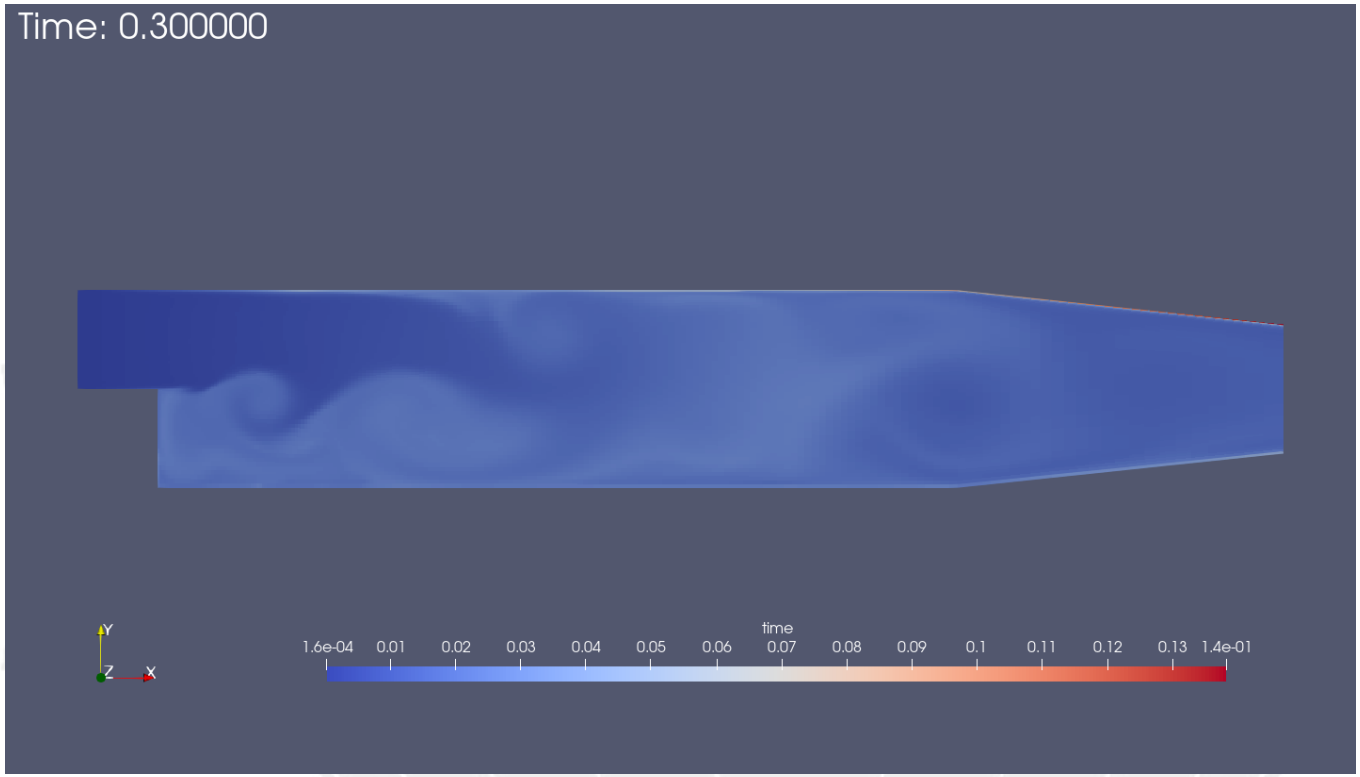


Figure: Flow time at the end of the simulation

Result: "converged" state reaction rate

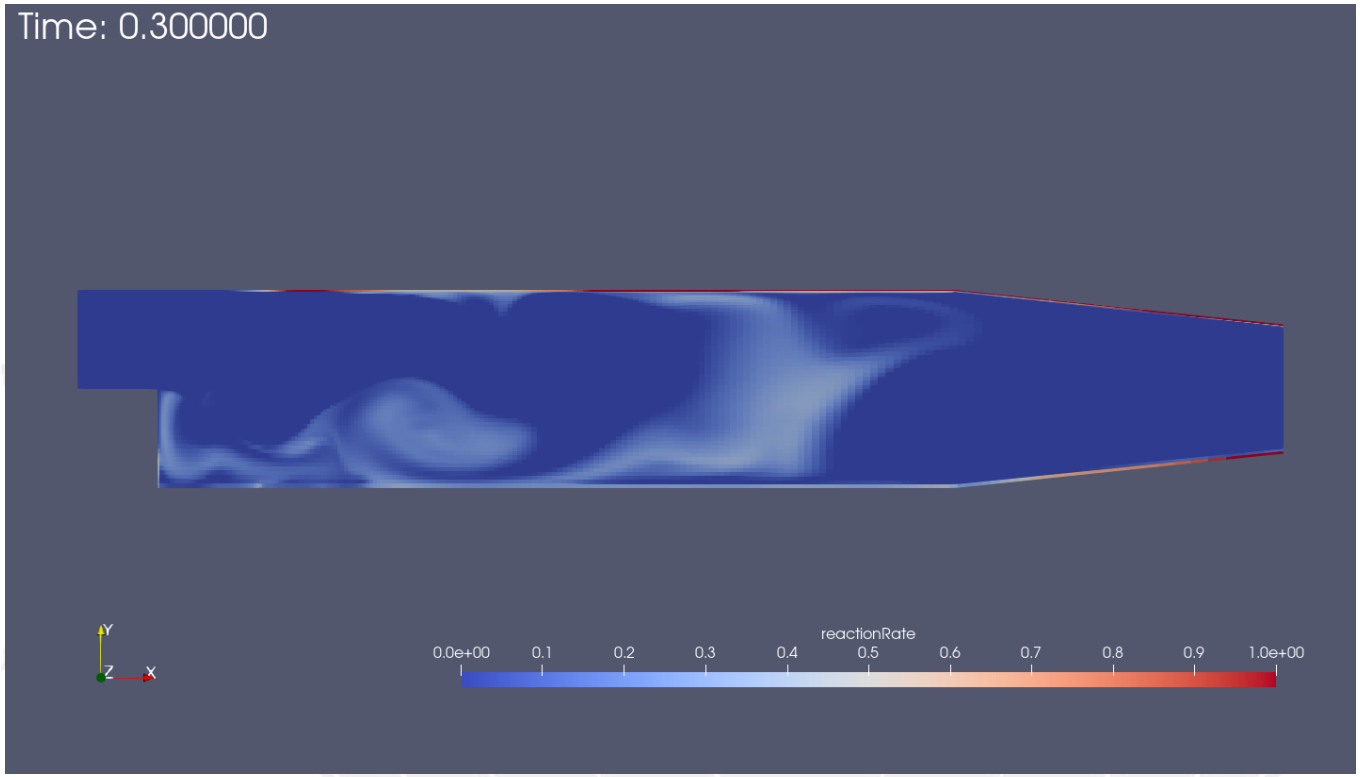


Figure: Reaction rate the end of the simulation

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

This was not Ingazs first try

- Usually what you see on the slides was not the first attempt to do it
  - This is the case here as well
- Ignaz first tried running the case without a "decay"

constant/timeTriggeredProperties

```
decay_speed [0 0 -1 -1 0 0 0] 0;
```

But this failed. Ignaz wants to understand why

- for this the data at the crash would be nice

Crash dumps for OpenFOAM

- Often a case crashes and it would be nice to see "why"
  - For that we'd need the field values at the crash time
- The `writeOldTimesOnSignal` function object does that
  - It "hooks" into OpenFOAMs signal handler
  - If one of the signals is raised it executes
    - Looks through the `objectRegistry` for fields
    - Writes them to disk
    - Executes the regular signal handler
- It tries to do that in parallel runs as well
  - Sometimes fails

"Storing" timesteps

- Crash dump is useful
  - but only shows the failed state
  - Usually the problem started before that
- For that a number of timesteps can be specified for which data should be stored
  - In case of a crash this data is written as well
- At every timestep the function object
  - searches memory for writeable fields
  - copies them
  - removes extra timestep data

Enabling crash dumps

functions

```
storeAndWriteOnCrash {
    type writeOldTimesOnSignal;
    numberOfTimestepsToStore 10;
    writeCurrent yes;
    sigFPE true;
    sigSEGV true;
    sigINT true;
    sigQUIT true;
    sigABRT true;
}
```

Parameters

Individual signals can be switched on and off

numberOfTimestepsToStore	Number of timesteps
writeCurrent	should the current state be saved
sigFPE	floating point exception
sigSEGV	segmentation fault
sigINT	case was stopped with `Ctrl-C`
sigABRT	OpenFOAM called `abort()`
sigQUIT	process was stopped with the `kill` command

Running to the crash

Ignaz runs till the crash

```
> pyFoamPlotRunner.py --with-all --progress --hardcopy --clear auto
 t =     0.05953
Killing PID 12999
 PyFoam WARNING on line 292 of file /home/bgschaid/Projects/PyFoam/src/PyFoam/Execution/<brk>
       <cont>FoamThread.py : Process 12999 was already dead
> ls
0      0.05943   0.05949   bound_.png     courant.png
0.01   0.05944   0.0595    bound.png      custom0000_time.png
0.02   0.05945   0.05951   constant       custom0001_temperature.png
0.03   0.05946   0.05952   cont_.png      custom0002_ReactionRate.png
0.04   0.05947   0.05953   cont.png       customRegexp
0.05   0.05948   0.org     courant_.png   execution_.png
```

The crash: results - temperature

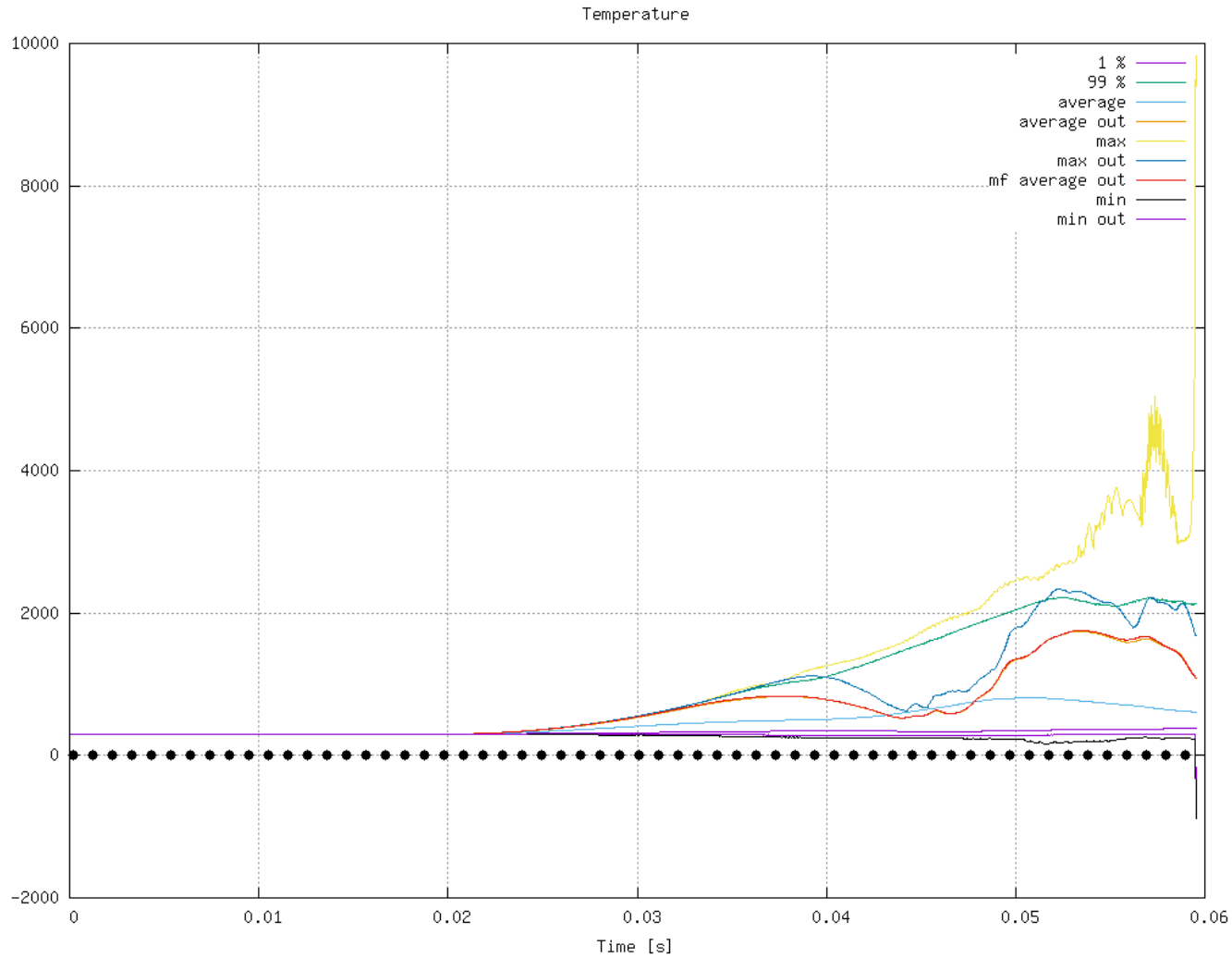


Figure: Temperature during the crashing run

The crash: results - flow time

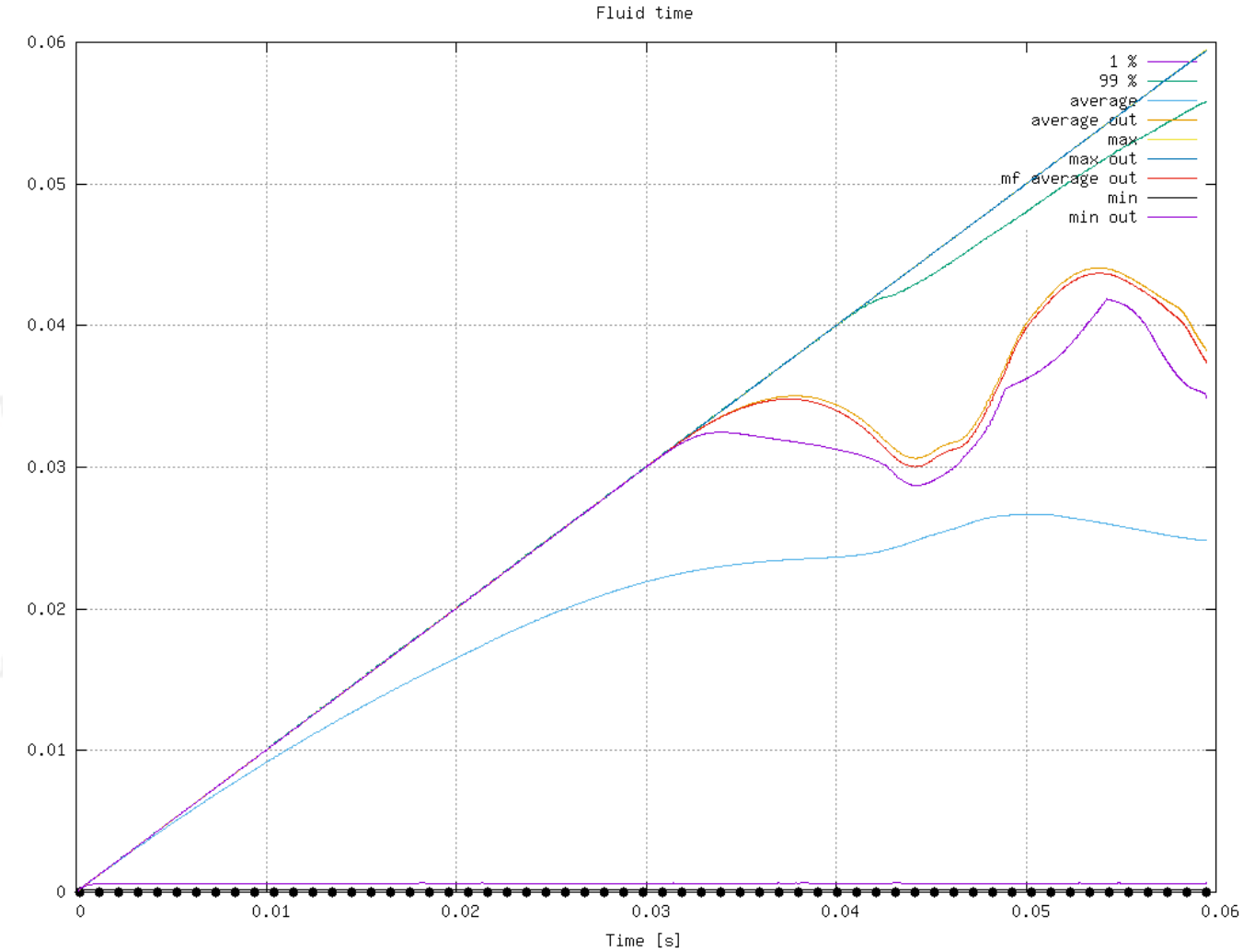


Figure: Flow time during the crashing run

The crash: results -temperature before crash

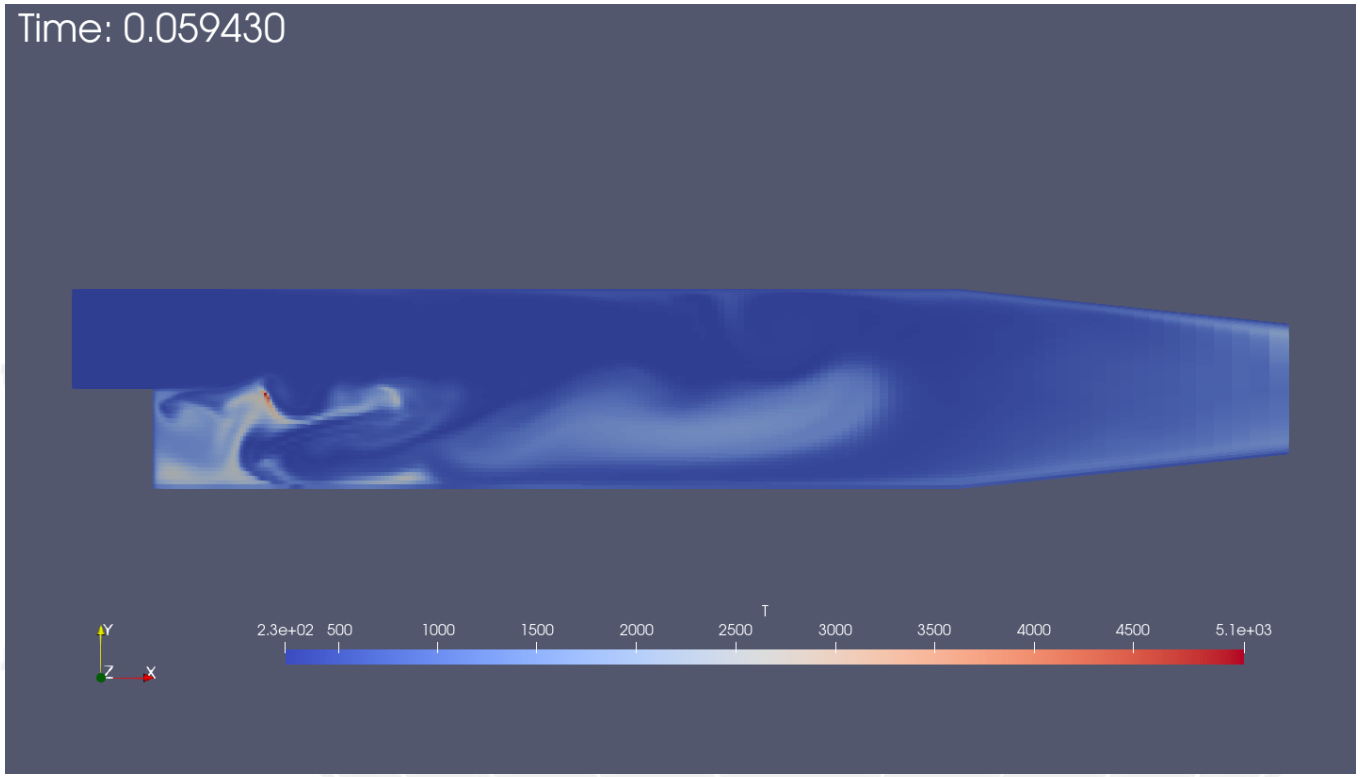


Figure: Temperature 10 time-steps before the crash

Beware: time and memory

This function object can be quite resource intensive

Memory Obviously if N timesteps are stored $N+1$ as much memory as in normal operation is needed

- Don't use it for large cases

Computation time Memory-bandwidth is the most expensive resource for CFD-simulations

- Field copy needs a lot of that
- Sometimes the function object needs 10% of the computation time

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

The high temperatures

- Ignaz noticed high temperature spikes during the simulation
  - Much higher than the "cut-off" temperature of 400
- He wants to investigate what is going on there
  - Adjusting `controlDict` manually to write at these times is tedious
    - Also error-prone
    - And uses a lot of computation time

Crashing the run with high temperatures

- One possibility:
  - Stop the run when the suspicious state occurs
    - In our case: high temperatures
  - Inspect the data
- The `writeAndEndSwakExpression;` function object allows that
  - Works with any condition
- But we'll let the case continue
  - But you can try it as an exercise

Writing by conditions

- The `writeIfSwakExpression` allows writing things depending on a condition
- At ever time a `writeCondition` is evaluated
  - `writeConditionAccumulation` tells where it should be true to trigger a write

    or in at least one cell
    and in all cells
- Storing and writing of times can be switched on
  - This allows investigating what led to the "event"

The different stages

The function object goes through different stages

1. Regular mode: nothing is written and the expression is evaluated
2. `write` mode: the condition has triggered
   - how long that takes is controlled by `writeControlMode`
3. `cooldown` mode: to avoid immediately going into `write` mode in this mode nothing is written
   - that is controlled by `cooldownMode`
4. goes back to regular mode

`write` and `cooldown` can be specified in different ways

- a swak expression
- a simulation time
- a number of timesteps

Write before and after

functions

```
writeLargeT {
    type writeIfSwakExpression;

    outputControlMode timeStep;
    outputInterval 1;

    writeControlMode timesteps;
    cooldownMode intervall;

    storeAndWritePreviousState true;
    numberOfTimestepsToStore 5;

    writeTimesteps 5;
    cooldownIntervall 0.001;

    valueType internalField;
    writeCondition "T>600";
    writeConditionAccumulation or;
}
```

The written times

More written times

```
> ls
0         0.03509   0.08      0.10886   0.11102   0.13411   0.17132   0.21328   0.25818   0.28414
0.01      0.0351    0.09      0.10887   0.11103   0.13412   0.17133   0.21329   0.25819   0.28415
0.02      0.03511   0.1       0.10888   0.11104   0.13413   0.17134   0.2133    0.2582    0.29
0.03      0.03512   0.10673   0.10889   0.11105   0.14      0.17304   0.22      0.25916   0.3
0.03299   0.03513   0.10674   0.1089    0.11106   0.15      0.17305   0.23      0.25917   0.org
0.033     0.03514   0.10675   0.10891   0.11292   0.16      0.17306   0.24      0.25918
0.03301   0.03515   0.10676   0.10892   0.11293   0.17      0.17307   0.25      0.25919
0.03302   0.03516   0.10677   0.10893   0.11294   0.17016   0.17308   0.25681   0.2592
0.03303   0.03517   0.10678   0.10894   0.11295   0.17017   0.17309   0.25682   0.25921
0.03304   0.03518   0.10679   0.10991   0.11296   0.17018   0.1731    0.25683   0.25922
0.03305   0.03946   0.1068    0.10992   0.11297   0.17019   0.17311   0.25684   0.25923
0.03306   0.03947   0.10681   0.10993   0.11298   0.1702    0.17312   0.25685   0.25924
0.03307   0.03948   0.10682   0.10994   0.11299   0.17021   0.17313   0.25686   0.25925
0.03308   0.03949   0.10779   0.10995   0.113     0.17022   0.18      0.25687   0.26
0.03404   0.0395    0.1078    0.10996   0.11301   0.17023   0.19      0.25688   0.27
0.03405   0.03951   0.10781   0.10997   0.12      0.17024   0.2       0.25689   0.28
0.03406   0.03952   0.10782   0.10998   0.13      0.17025   0.21      0.2569    0.28406
0.03407   0.03953   0.10783   0.10999   0.13404   0.17125   0.21321   0.25811   0.28407
0.03408   0.03954   0.10784   0.11      0.13405   0.17126   0.21322   0.25812   0.28408
0.03409   0.03955   0.10785   0.11097   0.13406   0.17127   0.21323   0.25813   0.28409
0.0341    0.04      0.10786   0.11098   0.13407   0.17128   0.21324   0.25814   0.2841
0.03411   0.05      0.10787   0.11099   0.13408   0.17129   0.21325   0.25815   0.28411
0.03412   0.06      0.10788   0.111     0.13409   0.1713    0.21326   0.25816   0.28412
0.03413   0.07      0.10885   0.11101   0.1341    0.17131   0.21327   0.25817   0.28413
```

Result: High temperature

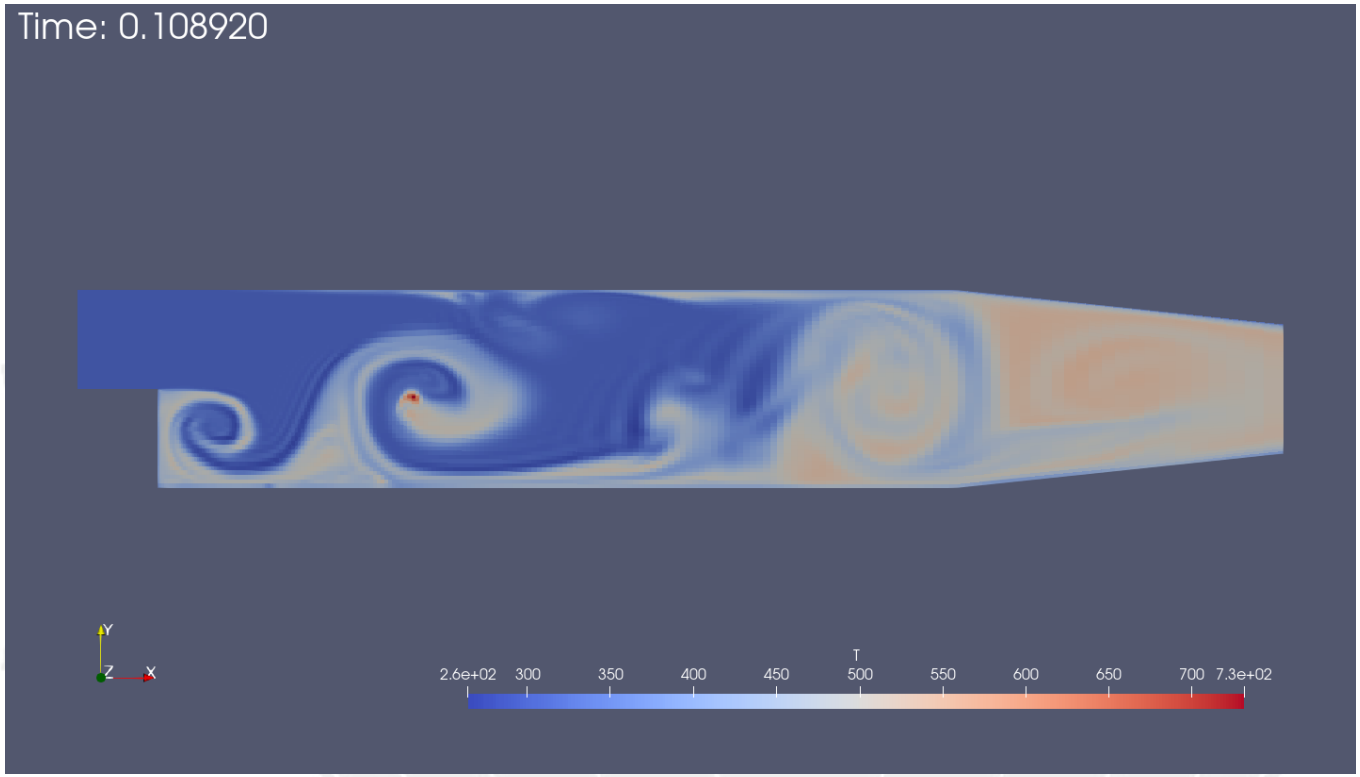


Figure: Temperature during a peak

Result: Reaction rate is not to blame

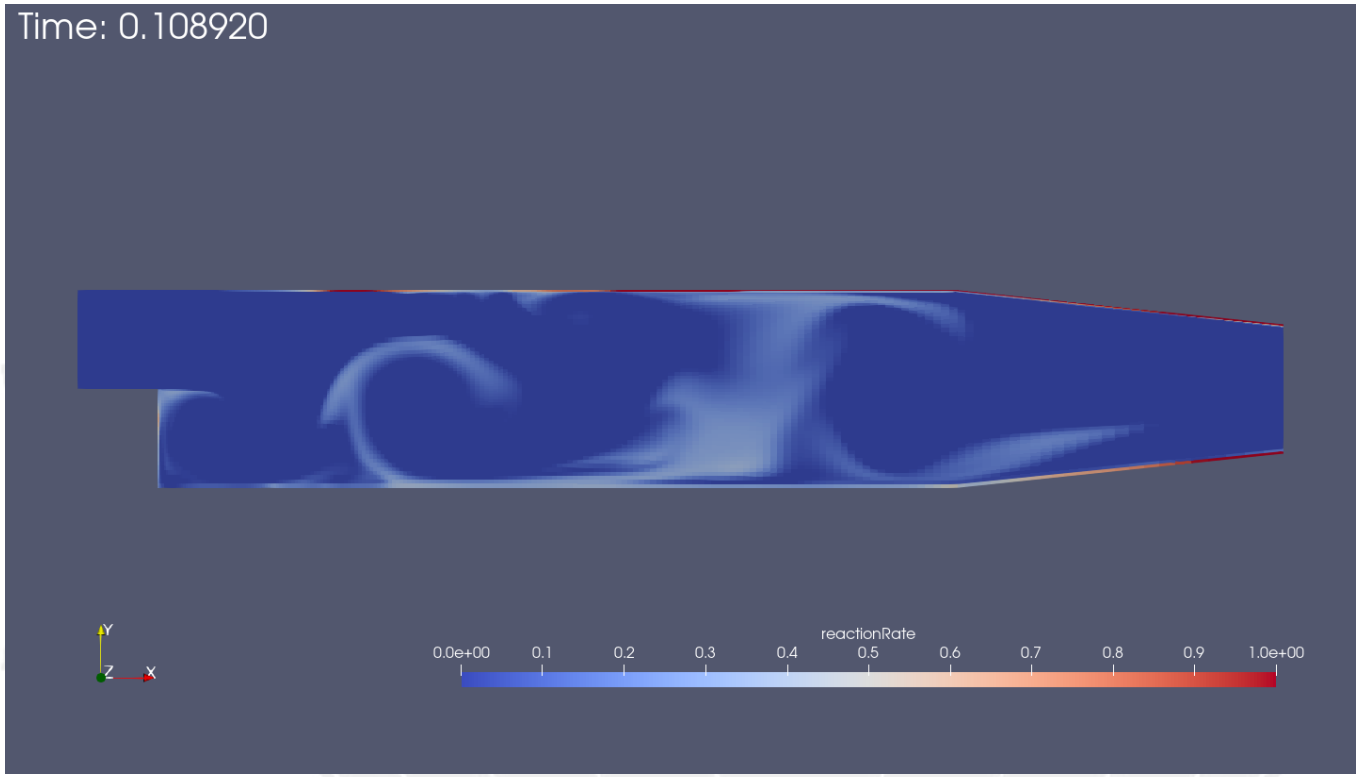


Figure: Reaction rate at the same time

Result: Flow time

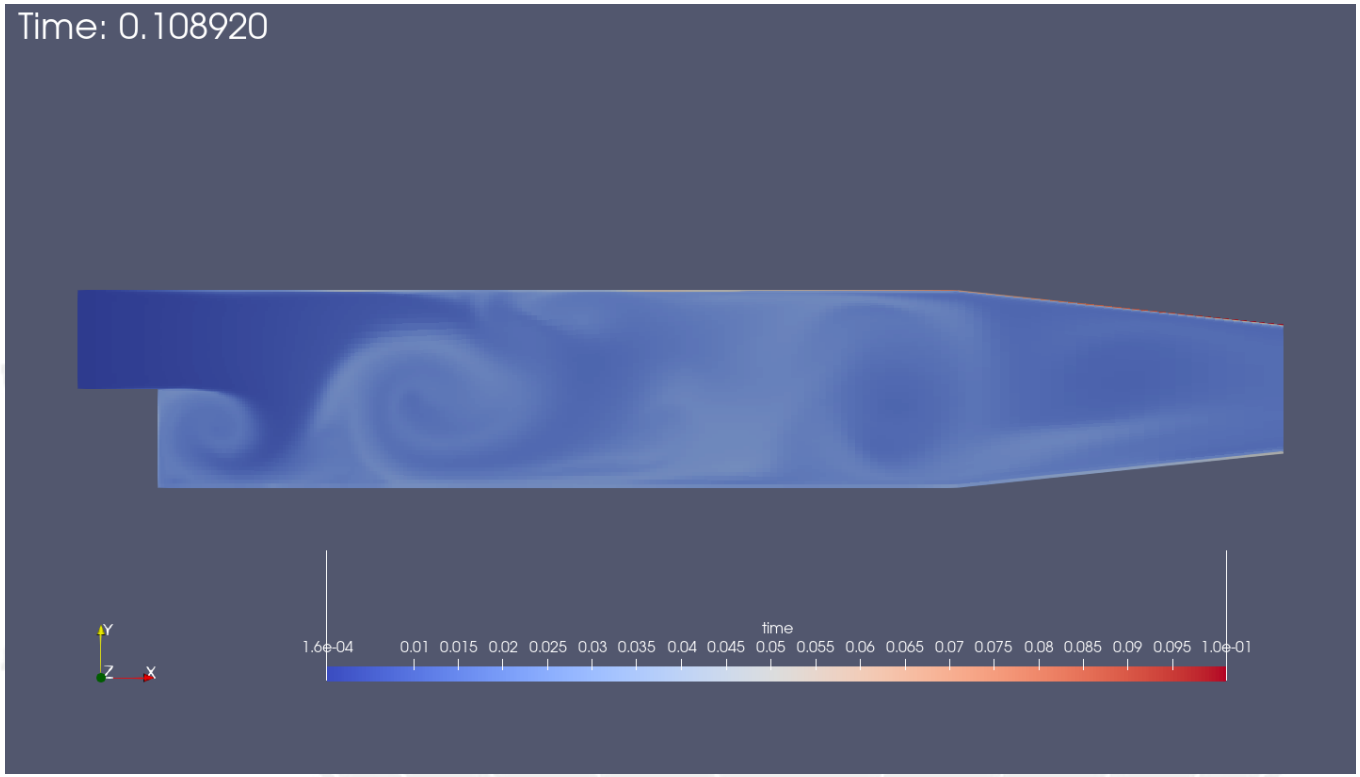


Figure: Flow time at the same time

Discretization?

- The high temperature does not correspond with a high reaction rate
  - So it does not seem to be a "model problem"
- Happens in a "difficult" region
  - Center of a vortex
  - Maybe the flow field there is not divergence free

Ignaz decides that more investigation in the used discretization schemes is needed

- swak4foam has plugin-functions for that

Beware: duplication

- When storing time-step data the discussed function objects do not share
  - Everyone has its own copy
  - This might be a memory problem
    - especially if multiple `writeIfSwakExpression` objects are used
- This is planned to be resolved in future swak4Foam-versions

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Which fvOptions are called

Which fvOption-hooks are present

- Sometimes you want to know "when are which fvOptions for which fields called"
  - This can be answered by diving into the code
- reportAvailableFvOptions prints that to the console
  - once you found the ones you want to "hook" into disable it

fvOptions

```
showFvOptions {
    type reportAvailableFvOptions;
    active true;
    selectionMode all;
    reportAvailableFvOptionsCoeffs {}
}
```

Using function objects on intermediate results

`fvOption` calls `function`

- Sometimes Ignaz wants to see information about intermediate values
  - Not just at the end of the time-step like regular function objects
- `executeFunctionObjectsFvOption` allows him to execute function objects whenever a fvOption can be called
  - fieldNames specifies the names of the fields to hook into
  - doXX specifies which kind of fvOption to execute it with
  - functions a dictionary with function objects
- every function object can be used
  - not all of them can handle this
  - it is possible to shoot yourself in the foot
    - but that is always possible

`fvOptions`

```
reportStuff {
    type executeFunctionObjectsFvOption;
    active true;
    selectionMode all;

    executeFunctionObjectsFvOptionCoeffs {
        fieldNames (
            time
        );

        //        verbose true;
        verbose false;

        doCorrect false;
        doAddSup true;
        doSetValue false;
        doMakeRelative false;
        doMakeAbsolute false;

        functions {
            excess_temperature {
                type swakExpression;
                valueType internalField;

                executeMoreThanOnce yes;
                //                noWrite <brk>
                      <cont>yes;
                noExecute yes;

                libs (<brk>
                      <cont>simpleSwakFunctionObjects<brk>
                      <cont>);
                verbose true;
                expression "<brk>
                      <cont>excess_temperature<brk>
                      <cont>";
                aliases {
                    flowTime time;
                }
                accumulations (
                    min
                    weightedQuantile0.01
                    weightedAverage
                    weightedQuantile0.99
                    max
                );
            }
            store_extra_temperature {
                type expressionField;
                libs (swakFunctionObjects);
                autowrite true;
                fieldName "extraTemperature<brk>
                      <cont>";
                expression "<brk>
                      <cont>excess_temperature<brk>
                      <cont>";
```

Getting "residual" fields

- There are `fvOptions` in swak that allow calculating the "residual field" for an equation
- This is explained in the "State and solution" presentation from 2017 in Exceter

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

Ignaz has to justify the many function objects

- Ignaz likes to add swak function objects
  - Sometimes there is more "swak output" on the console than "solver output"
  - People who see that suggest "you're making your simulations 2 times slower"
  - Ignaz has a hard time suggesting to people that the p-solver only generates one line of output but uses most of the computation time
- So he routinely uses *high-level profiling* to be able to tell people "see: all the function objects only use 5% of the time"
  - Worth it: they produce results. And results is what CFD is about

High-level vs low-level profiling

- Usually when we talk about profiling we talk about *low-level* profiling
  - The code is compiled in a special way (for debug-symbols)
  - The program is executed inside a special program (`valgrind` for instance)
  - That program records how often each instruction is executed
  - Advantage: very detailed information about where the time is spent
  - Disadvantage: makes execution much slower
- High-level profiling
  - The user "instruments" the code
    - Adds special instructions "how long does this code section take"
  - the recorded info is written in the end
  - If done correctly almost no performance impact

Profiling support in OpenFOAM-versions

This kind of profiling does not exist in all OpenFOAM-versions

Foundation release search the message board for "Feature proposal: Application level profiling" for an explanation why the don't include it

foam-extend accepted and included it long ago

ESI release accepted and included it. Refactored it since then

If activated the profiling information is written to `uniform/profiling` at each time-step

Switching it on in v2012

Profiling is not activated "out of the box" in the ESI-release

controlDict

```
profiling
{
    active      true;
    cpuInfo     true;
    memInfo     true;
    sysInfo     true;
}
```

What is profiled "out of the box"

To make it useful some code sections are instrumented "out of the box"

- time-loop
- linear solvers (that should use the most time)
- writing of the values
- function object execution

Profiling triggers

- Profiling is implemented by "triggers"
  - Little objects
    - Construction starts the clock
    - Destruction stops the clock
    - ... or the clock is stopped manually
  - So scoping is important
- Triggers are organized in a stack
  - At creation the are added
  - At destruction removed
  - A trigger is the "child" of the trigger immediately above on the stack
- Triggers also record how often they were triggered
- When creating a trigger a "C++ name" and a name that is shown in the information are specified

Adding profiling

Ignaz adds two triggers

- timeEqn is stopped by the scoping rules of C++
- excTemp is stopped manually

timeEqn.H

```
{
    addProfiling(timeEqn, "timeEqn.H");

    addProfiling(excTemp, "excess_temperature calculation");
    volScalarField excess_temperature(
        "excess_temperature",
        T-decay_temperature
    );

    forAll(excess_temperature, cellI) {
        if(excess_temperature[cellI] < 0) {
            excess_temperature[cellI] = 0;
        }
    }
    excess_temperature.correctBoundaryConditions();

    endProfiling(excTemp);
```

"Child" vs "Own" time

- Every trigger knows its "children"
- So it knows the time spent in the children
- Time not spent in the children is called "own time"
  - If that is a high percentage then more detailed profiling might be of interest

Where not to add profiling

Where not to add triggers

- Recording the time doesn't take much time
  - But this can add up if done often
- So don't add this to the inner-most loop
- Rules of thumb:
  - Think 3 times before adding it inside a `forAll`
  - Don't add it to functions that work on a single cell/face/point
- In these cases you'll want low-level profiling

Bad

```
forAll(excess_temperature, cellI) {
    addProfiling(exc, "in loop");

    if(excess_temperature[cellI] < 0) {
        excess_temperature[cellI] = 0;
    }
}
```

Non-developing uses

- This profiling is also interesting for non-developers
- To optimize the run-time of the cases
  - Answer questions like "Are 123 iterations of a CG-solver faster or slower than 5 AMG-iterations"
- For this the out-of-the-box triggers are sufficient

Instrumented solver

The sources of the instrumented solver are in `04rhoTimeTriggerFoamProfile.tar.gz` (or `/Examples` of the docker image. You know the drill)

Ignaz is not happy

- The profiling output is hard to read
- But Ignaz knows a utility for this

uniform/profiling

```
profiling
{
    trigger0
    {
        id              0;
        description     "application::main";
        calls           1;
        totalTime       391.19;
        childTime       390.923;
        active          true;
    }

    trigger17
    {
        id              17;
        parentId        0;
        description     "time.run()␣timeTriggeredPitzDaily";
        calls           4000;
        totalTime       369.042;
        childTime       362.982;
        maxMem          382884;
        active          true;
    }

    trigger1
    {
        id              1;
```

Outline

1 Introduction
- This presentation
- Who is this?
- What are we working with
- Before we start
- What we simulate

2 Setup
- OpenFOAMs developer support
- PyFoams helps

3 Prototyping
- Solving simple PDEs
- Other

4 Development
- The solver
- What happened
- Irregular conditions
- fvOptions

5 Optimizing
- High-level profiling
- Pyfoam support for profiling output

6 Conclusion

The utility

- PyFoam has a utility to deal with the profiling output
  - Reads and analyzes it
  - Prints it in a more readable form
  - generates a graph
- Usage examples are printed with the `--help`-option

Example output

Profiling data

```
> pyFoamListProfilingInfo.py . --time=0.05
Profiling info from time 0.05
                                                                                |  total  ( self ) - parent ( self ) |    calls      total       self
--------------------------------------------------------------------------------------------------------------------------------------------------
application::main                                                               |  100.0% (  0.1%) - 100.0% (  0.1%) |        1       491s      0.307s
|- functionObjects::read                                                        |    0.0% (  0.0%) -   0.0% (  2.4%) |        1    0.04492s   0.001059s
  |- functionObject::excess_temperature::new                                    |    0.0% (  0.0%) -  51.8% (100.0%) |        1    0.02325s    0.02325s
  |- functionObject::store_extra_temperature::new                               |    0.0% (  0.0%) -  45.9% (100.0%) |        1    0.02061s    0.02061s
|- fvOption::correct.showFvOptions                                              |    0.0% (  0.0%) -   0.0% (100.0%) |        1 6.049e-05s 6.049e-05s
|- functionObjects.start()                                                      |    0.0% (  0.0%) -   0.0% (  0.6%) |        1    0.03375s   0.0001871s
  |- functionObjects::read                                                      |    0.0% (  0.0%) -  99.4% (  0.4%) |        1    0.03356s   0.0001487s
    |- functionObject::fieldAverage1::new                                       |    0.0% (  0.0%) -  93.3% (100.0%) |        1    0.03131s    0.03131s
    |- functionObject::timeStatistics::new                                      |    0.0% (  0.0%) -   1.7% (100.0%) |        1 0.0005854s 0.0005854s
    |- functionObject::temperatureStatistics::new                               |    0.0% (  0.0%) -   0.7% (100.0%) |        1 0.0002407s 0.0002407s
    |- functionObject::reactionRateStatistics::new                              |    0.0% (  0.0%) -   0.7% (100.0%) |        1  0.000237s  0.000237s
    |- functionObject::timeOutStatistics::new                                   |    0.0% (  0.0%) -   0.8% (100.0%) |        1 0.0002532s 0.0002532s
    |- functionObject::temperatureOutStatistics::new                            |    0.0% (  0.0%) -   0.6% (100.0%) |        1 0.0002171s 0.0002171s
    |- functionObject::timeOutMassFlowAveraged::new                             |    0.0% (  0.0%) -   0.7% (100.0%) |        1 0.0002501s 0.0002501s
    |- functionObject::temperatureOutMassFlowAveraged::new                      |    0.0% (  0.0%) -   0.7% (100.0%) |        1 0.0002204s 0.0002204s
    |- functionObject::writeEnergy::new                                         |    0.0% (  0.0%) -   0.1% (100.0%) |        1 4.829e-05s 4.829e-05s
    |- functionObject::storeAndWriteOnCrash::new                                |    0.0% (  0.0%) -   0.2% (100.0%) |        1 5.285e-05s 5.285e-05s
|- time.run() timeTriggeredPitzDaily                                            |   94.4% (  1.5%) -  94.4% (  1.6%) |     5000     463.5s      7.602s
  |- fvOption().showFvOptions                                                   |    0.1% (  0.1%) -   0.1% (100.0%) |     5000    0.2924s     0.2924s
  |- fvOption::constrain.rho                                                    |    0.0% (  0.0%) -   0.0% (100.0%) |     5000     0.201s      0.201s
  |- fvMatrix::solve.rho                                                        |    0.2% (  0.1%) -   0.2% ( 67.4%) |     5000     0.9678s     0.6523s
    |- lduMatrix::solver.rho                                                    |    0.1% (  0.1%) -  32.6% (100.0%) |     5000     0.3155s     0.3155s
  |- fvOption::correct.showFvOptions                                            |    0.0% (  0.0%) -   0.0% (100.0%) |     5000     0.2062s     0.2062s
  |- UEqn.H                                                                     |   28.8% ( 22.1%) -  30.5% ( 76.7%) |    15000     141.5s      108.5s
    |- fvOption().showFvOptions                                                 |    0.3% (  0.3%) -   0.9% (100.0%) |    15000      1.29s       1.29s
    |- fvOption::constrain.U                                                    |    0.1% (  0.1%) -   0.5% (100.0%) |    15000     0.6878s     0.6878s
    |- fvMatrix::solve.U                                                        |    6.2% (  1.4%) -  21.4% ( 23.5%) |    15000     30.26s      7.098s
      |- lduMatrix::solver.Ux                                                   |    2.3% (  1.0%) -  36.8% ( 45.7%) |    15000     11.12s      5.08s
        |- lduMatrix::smoother.Ux                                               |    1.2% (  1.2%) -  54.3% (100.0%) |    12845     6.044s      6.044s
      |- lduMatrix::solver.Uy                                                   |    2.5% (  1.0%) -  39.8% ( 40.5%) |    15000     12.04s      4.872s
        |- lduMatrix::smoother.Uy                                               |    1.5% (  1.5%) -  59.5% (100.0%) |    14990     7.165s      7.165s
    |- fvOption::correct.showFvOptions                                          |    0.2% (  0.2%) -   0.5% (100.0%) |    15000     0.737s      0.737s
  |- timeEqn.H                                                                  |   15.0% (  2.6%) -  15.9% ( 17.6%) |    15000     73.62s      12.99s
    |- excess_temperature calculation                                           |    0.2% (  0.2%) -   1.4% (100.0%) |    15000      1.06s       1.06s
    |- fvOption().showFvOptions                                                 |    0.1% (  0.1%) -   1.0% (100.0%) |    15000     0.7155s     0.7155s
    |- fvOption().matrixReport                                                  |    0.1% (  0.1%) -   0.6% (100.0%) |    15000     0.4751s     0.4751s
    |- fvOption().timeSourceBefore                                              |    0.5% (  0.5%) -   3.2% (100.0%) |    15000     2.356s      2.356s
    |- fvOption().timeResidual                                                  |    0.1% (  0.1%) -   0.9% (100.0%) |    15000     0.6331s     0.6331s
    |- fvOption().timeResidual2                                                 |    0.4% (  0.4%) -   2.8% (100.0%) |    15000      2.06s       2.06s
    |- fvOption().reportStuff                                                   |    1.6% (  0.1%) -  10.9% (  8.2%) |    15000     8.023s      0.6599s
      |- functionObject::excess_temperature::execute                            |    0.1% (  0.1%) -   5.9% (100.0%) |    15000     0.4773s     0.4773s
      |- functionObject::excess_temperature:write                               |    1.4% (  1.4%) -  85.8% (100.0%) |    15000     6.886s      6.886s
    |- fvOption().timeSourceAfter                                               |    0.1% (  0.1%) -   1.0% (100.0%) |    15000     0.7279s     0.7279s
    |- fvOption::constrain.time                                                 |    5.9% (  5.9%) -  39.3% (100.0%) |   105000     28.91s      28.91s
```

Restricting output

Ignaz is not interested in the "small timers"

Skipping items with less than 1 percent

```
> pyFoamListProfilingInfo.py . --time=0.05 --threshold-low=1
Profiling info from time 0.05
                                                                  |  total  ( self ) - parent ( self ) |    calls      total      self
------------------------------------------------------------------------------------------------------------------------------------
application::main                                                 |  100.0% (  0.1%) - 100.0% (  0.1%) |        1      491s     0.307s
|- time.run() timeTriggeredPitzDaily                              |   94.4% (  1.5%) -  94.4% (  1.6%) |     5000    463.5s     7.602s
  |- UEqn.H                                                       |   28.8% ( 22.1%) -  30.5% ( 76.7%) |    15000    141.5s     108.5s
    |- fvMatrix::solve.U                                          |    6.2% (  1.4%) -  21.4% ( 23.5%) |    15000    30.26s     7.098s
      |- lduMatrix::solver.Ux                                     |    2.3% (  1.0%) -  36.8% ( 45.7%) |    15000    11.12s      5.08s
        |- lduMatrix::smoother.Ux                                 |    1.2% (  1.2%) -  54.3% (100.0%) |    12845    6.044s     6.044s
      |- lduMatrix::solver.Uy                                     |    2.5% (  1.0%) -  39.8% ( 40.5%) |    15000    12.04s     4.872s
        |- lduMatrix::smoother.Uy                                 |    1.5% (  1.5%) -  59.5% (100.0%) |    14990    7.165s     7.165s
    |- < less than 1.0% >                                         |    0.6% (  0.6%) -   1.9% (100.0%) |    45000    2.714s     2.714s
  |- timeEqn.H                                                    |   15.0% (  2.6%) -  15.9% ( 17.6%) |    15000    73.62s     12.99s
    |- fvOption().reportStuff                                     |    1.6% (  0.1%) -  10.9% (  8.2%) |    15000    8.023s    0.6599s
      |- functionObject::excess_temperature:write                 |    1.4% (  1.4%) -  85.8% (100.0%) |    15000    6.886s     6.886s
      |- < less than 1.0% >                                       |    0.1% (  0.1%) -   5.9% (100.0%) |    15000   0.4773s    0.4773s
    |- fvOption::constrain.time                                   |    5.9% (  5.9%) -  39.3% (100.0%) |   105000    28.91s     28.91s
    |- fvMatrix::solve.time                                       |    2.3% (  0.5%) -  15.4% ( 21.2%) |    15000    11.31s     2.392s
      |- lduMatrix::solver.time                                   |    1.8% (  0.9%) -  78.8% ( 51.8%) |    15000    8.915s     4.615s
        |- < less than 1.0% >                                     |    0.9% (  0.9%) -  48.2% (100.0%) |     9243     4.3s       4.3s
    |- < less than 1.0% >                                         |    2.5% (  2.5%) -  16.8% (100.0%) |   225000    12.39s     12.39s
  |- EEqn.H                                                       |   13.6% ( 10.2%) -  14.4% ( 74.9%) |    15000    66.88s     50.07s
    |- fvMatrix::solve.e                                          |    2.9% (  0.5%) -  21.4% ( 17.9%) |    15000     14.3s     2.557s
      |- lduMatrix::solver.e                                      |    2.4% (  0.9%) -  82.1% ( 39.5%) |    15000    11.75s     4.643s
        |- lduMatrix::smoother.e                                  |    1.4% (  1.4%) -  60.5% (100.0%) |    15000    7.102s     7.102s
    |- < less than 1.0% >                                         |    0.5% (  0.5%) -   3.7% (100.0%) |    45000    2.507s     2.507s
  |- pEqn.H                                                       |   29.5% ( 14.1%) -  31.2% ( 47.9%) |    15000    144.8s     69.37s
    |- fvMatrix::solve.p                                          |   13.9% (  0.7%) -  47.1% (  5.0%) |    15000    68.24s     3.408s
      |- lduMatrix::solver.p                                      |   13.2% ( 13.2%) -  95.0% (100.0%) |    15000    64.84s     64.84s
    |- < less than 1.0% >                                         |    1.5% (  1.3%) -   4.9% ( 87.2%) |    90000     7.16s     6.247s
  |- turbulence->correct()                                        |    5.6% (  4.4%) -   5.9% ( 78.8%) |     5000    27.46s     21.65s
    |- < less than 1.0% >                                         |    1.2% (  0.4%) -  21.2% ( 32.3%) |    25000    5.809s     1.879s
  |- < less than 1.0% >                                           |    0.3% (  0.3%) -   0.4% ( 81.1%) |    20005    1.668s     1.353s
|- functionObjects.execute()                                      |    5.5% (  0.1%) -   5.5% (  1.3%) |     4999    27.11s    0.3546s
  |- functionObject::fieldAverage1::execute                       |    1.2% (  1.2%) -  21.3% (100.0%) |     4999    5.763s     5.763s
  |- functionObject::storeAndWriteOnCrash::execute                |    1.8% (  1.8%) -  31.9% (100.0%) |     4999    8.651s     8.651s
  |- < less than 1.0% >                                           |    2.5% (  2.5%) -  45.5% (100.0%) |    89982    12.34s     12.34s
|- < less than 1.0% >                                             |    0.0% (  0.0%) -   0.0% (  1.7%) |        3   0.07873s  0.001306s
```

Generating graphs

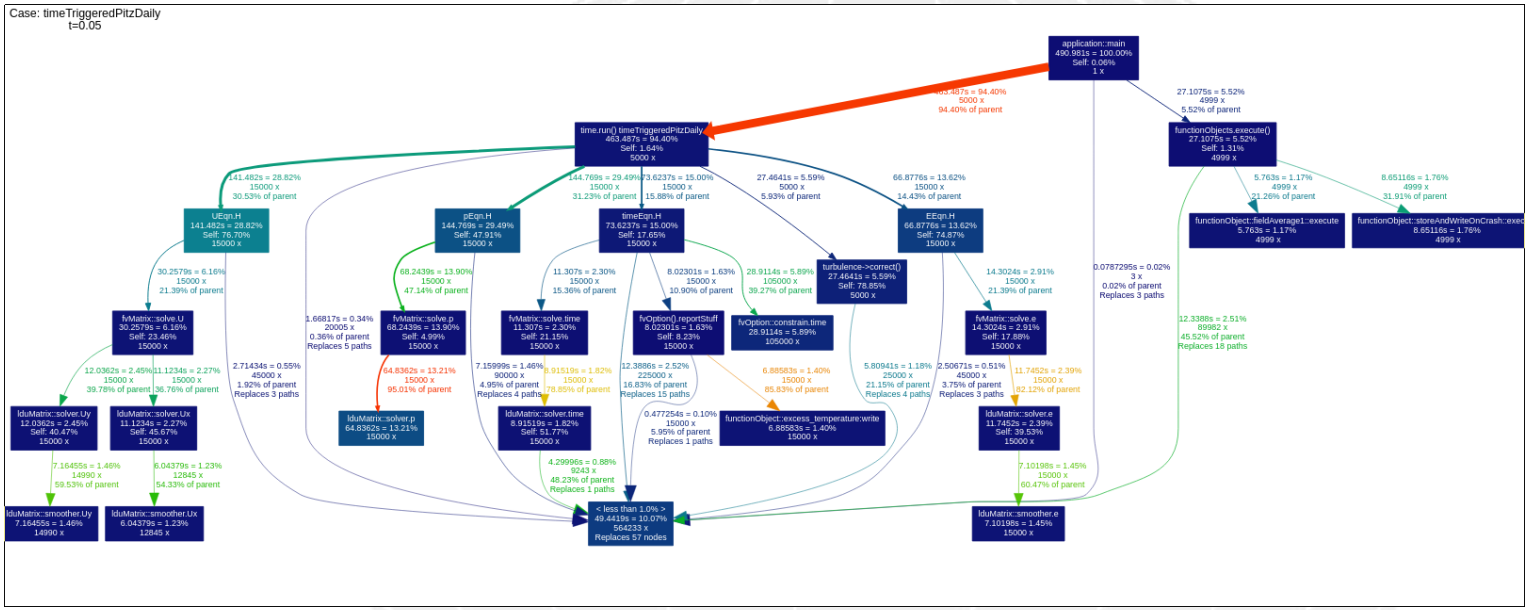
The utility can generate generate graph specifications for the dot-utility of the GraphViz-suite

- Just "pipe" into the utility
  - Specify options of the utility for different output formats
- "Boxes" are colored according to the own times
- "Arrow" thickness corresponds to the child time

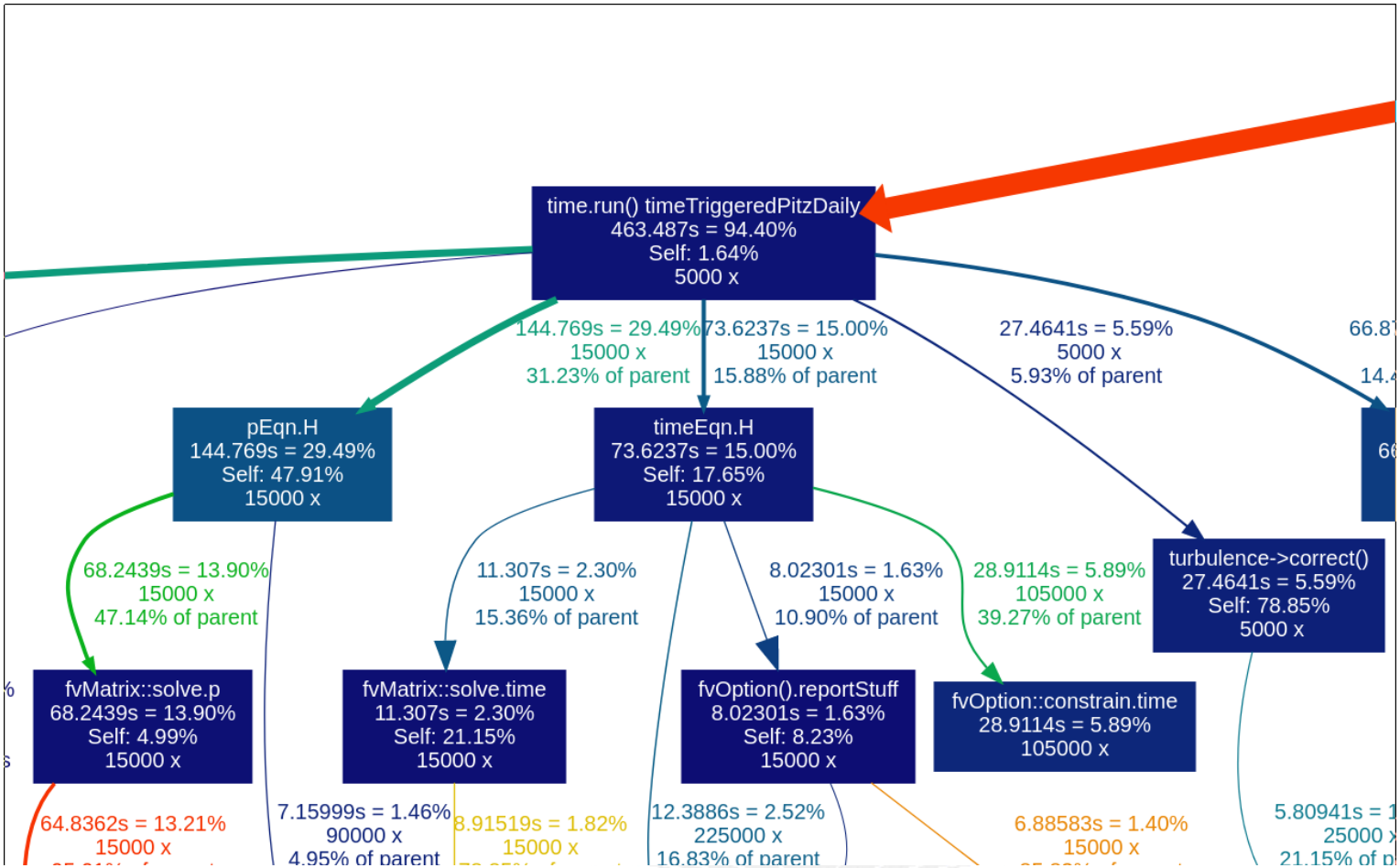
Generating a graph

```
> pyFoamListProfilingInfo.py . --time=0.05 --threshold-low=1 --graphviz-dot | dot -Tsvg -o <brk>
      <cont>timeTrigger.svg
> xdg-open timeTrigger.svg
```

An example graph



Zooming in


time.run() timeTriggeredPitzDaily
463.487s = 94.40%
Self: 1.64%
5000 x
144.769s = 29.49%
15000 x
31.23% of parent
73.6237s = 15.00%
15000 x
15.88% of parent
27.4641s = 5.59%
5000 x
5.93% of parent
pEqn.H
144.769s = 29.49%
Self: 47.91%
15000 x
timeEqn.H
73.6237s = 15.00%
Self: 17.65%
15000 x
68.2439s = 13.90%
15000 x
47.14% of parent
11.307s = 2.30%
15000 x
15.36% of parent
8.02301s = 1.63%
15000 x
10.90% of parent
28.9114s = 5.89%
105000 x
39.27% of parent
turbulence->correct()
27.4641s = 5.59%
Self: 78.85%
5000 x
fvMatrix::solve.p
68.2439s = 13.90%
Self: 4.99%
15000 x
fvMatrix::solve.time
11.307s = 2.30%
Self: 21.15%
15000 x
fvOption().reportStuff
8.02301s = 1.63%
Self: 8.23%
15000 x
fvOption::constrain.time
28.9114s = 5.89%
105000 x
64.8362s = 13.21%
15000 x
7.15999s = 1.46%
90000 x
4.95% of parent
8.91519s = 1.82%
15000 x
12.3886s = 2.52%
225000 x
16.83% of parent
6.88583s = 1.40%
15000 x
5.80941s = 1
25000 x
21.15% of p

Outline

1. Introduction
   - This presentation
   - Who is this?
   - What are we working with
   - Before we start
   - What we simulate
2. Setup
   - OpenFOAMs developer support
   - PyFoams helps
3. Prototyping
   - Solving simple PDEs
   - Other
4. Development
   - The solver
   - What happened
   - Irregular conditions
   - fvOptions
5. Optimizing
   - High-level profiling
   - Pyfoam support for profiling output
6. Conclusion

Ignaz says goodbye

- Ignaz hopes that you learned
  - How to use swak4Foam and PyFoam for your development work
  - Especially
    - Rapidly switching between versions
    - Getting "crash dumps" for cases
    - Find sections in the solver that use much time and need improvement
- He now gets back to his "combustion model"
  - And asks you not to reference/use it because
    - it is unphysical
    - it is proprietary to his company

Further presentations

- An introductory presentation about PyFoam and swakFoam was held yesterday
- At the workshop 2014 in Zagreb there was a "swak4Foam for Programmers" presentation which covers some things not covered here
- `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
- PyFoam has lots of ways to generate additional data (which might be helpful to debug the case)
  - 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
- A presentation "Expressive swak4Foam" about obscure corners of swak4Foam (was held in Duisburg 2019)
- A presentation "Programming with PyFoam" that was held at the 2020 Workshop

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