Virtual Tracer Tests: Coupling CFD and CREng to Simulate WRRFs Unit Processes

Introducing OpenFOAM®

nelson.marques@fsdynamics.pt; bruno.santos@fsdynamics.pt

1st September 2019

Watermatex 2019
DTU
Danmarks Tekniske Universitet
LUND UNIVERSITY
IWA the international water association
1 – 4 September 2019 | Copenhagen - Denmark
WATERMATEX 2019
10th IWA Symposium on Modelling and Integrated Assessment
Photo: Jacob Schjørring and Simone Lau, Copenhagen Media Center

OpenFOAM®

What it is and how to get it

Section Contents

What is OpenFOAM®? (1/3)

- OpenFOAM® is essentially an open-source software package that is primarily meant to be used as toolbox for applying the principles, methods and modelling strategies conceived in the field of Computational Fluid Dynamics.
- The acronym FOAM stands for "Field Operation and Manipulation".
- It is maintained and delivered by the OpenFOAM Foundation: www.openfoam.org
- OPENFOAM and OpenCFD are registered trademarks of OpenCFD Ltd (ESI Group) and also distribute their own builds: www.openfoam.com
- OpenFOAM as an open-source software package, is licensed under the GNU General Public License v3 (GPLv3): www.gnu.org/licenses/gpl.html

What is OpenFOAM®? (2/3)

- Users are free to use OpenFOAM software, which can be freely used and modified by each user in any field (personal, academic or commercial), without any licensing fees, as long as GPLv3 license terms are respected.
- The modifications to the source code only have to be made available to whom the binary packages are provided.
- In many simulation scenarios, OpenFOAM is ready to be used after installing.
- Nonetheless, not all modelling strategies are available out-of-the-box and the user may have to code a new modelling strategy, or deploy one already made available by the community that uses OpenFOAM.

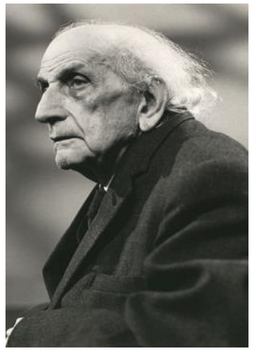
What is OpenFOAM®? (3/3)

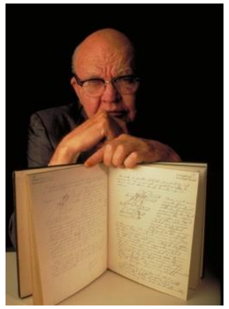
a

b

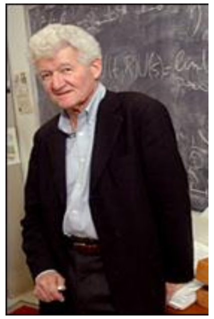
c

d

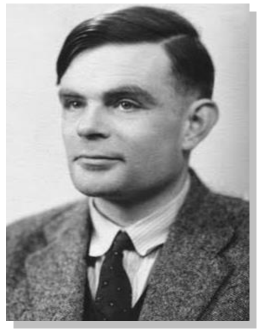

e

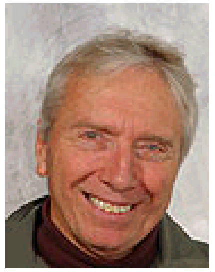

i

Courtesy of Texas Instruments


h

j


k


g

f

a) Anthony Jameson; b) Milovan Péric; c) Brian Spalding; d) Donald Knuth; e) Cornelius Lanczos, 1893-1974; f) Robert MacCormack; g) Brian Launder; h) Peter Lax; i) Jack Kilby, 1923-2005; j) John von Neumann, 1903-1957; k) Alan Turing, 1912-1954.

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Short history of OpenFOAM® (1/3)

- The original FOAM software was created by Henry Weller in 1989.
- Development of FOAM was done in an academic environment until 2000, including collaborative development.
- FOAM was commercialized as a CFD source code toolbox between 2000 and 2004 by the company Nabla Ltd.
- After the closure of Nabla Ltd in 2004, FOAM was modified, improved and released as open-source by OpenCFD on the 10th of December 2004, with the new name "OpenFOAM".
- The trade marks OPENFOAM and OPENCFD was registered ~2 years later, to help deter any abuse.

Short history of OpenFOAM® (2/3)

- OpenCFD was bought by SGI in 2011 and the OpenFOAM Foundation was created at the same time.
- The Foundation was created to ensure the source code remains open-source and the copyright is respected, independently of the trade mark.
- OpenCFD was later bought by ESI in 2012.
- In 2014, Henry Weller left OpenCFD/ESI and remains as director of the Foundation.
- 2015-17: Development in OpenFOAM continues to evolve, done by those collaborating with the Foundation.
- 2016-17: OpenCFD/ESI deliver their own development line too, with the alias OpenFOAM+, integrating changes by the Foundation.

Short history of OpenFOAM® (3/3)

- Although we have mostly mentioned Henry Weller as the original author, there have been a lot of contributions from several people and companies that have worked directly with him throughout FOAM/OpenFOAM's life span.
- Contributions are welcome and guidelines are outlined here:
  - openfoam.org/dev/how-to-contribute/
  - www.openfoam.com/community/repository.php
- References:
  - http://cfd.direct/openfoam/
  - http://www.openfoam.com/news/

Ecosystem around OpenFOAM® (1/3)

- The community that uses the technology mostly use this forum:
  - www.cfd-online.com/Forums/openfoam/
- The unofficial wiki, driven by the community: openfoamwiki.net
- The main public open-source forks of OpenFOAM:
  - foam-extend (foam-extend.org) is a community driven fork of OpenFOAM, mostly developed by Wikki Ltd: wikki.co.uk
  - Caelus-CML is another fork of OpenFOAM done by Applied CCM: www.caelus-cml.com

Ecosystem around OpenFOAM® (2/3)

- There are several variants of OpenFOAM, where most were created for adding support into the source code for working in other Operating Systems (Windows and Mac OS X).
- Complete list of forks and variants: openfoamwiki.net/index.php/Forks_and_Variants
- List of available forks/variants for Windows: http://openfoamwiki.net/index.php/Windows
- List of available forks/variants for Mac OS X: openfoamwiki.net/index.php/Installation/Mac_OS

Ecosystem around OpenFOAM® (3/3)

- Major contributions done by the community as toolboxes:
  - PyFoam is a Python based scripting toolkit, which enhances the abilities for using OpenFOAM from the command line: openfoamwiki.net/index.php/Contrib/PyFoam
  - swak4Foam is a toolkit designed for users that don't know C++, making it easier to use simple mathematical code in utilities, boundary conditions and post-processing tools: openfoamwiki.net/index.php/Contrib/swak4Foam
- All known community contributions independent of OpenFOAM:
  - openfoamwiki.net/index.php/Contrib
  - openfoamwiki.net/index.php/Extend-bazaar

Operating Systems and Standards (1/4)

- Before 1980, one of the most common operating system (OS) was Unix, of which there were several variants, most incompatible with each other.
- In 1981 MS-DOS was released, which was completely incompatible with Unix systems, but was easier to use.
- The first Mac OS was released in 1984, an alternative to all other operating systems.
- Microsoft Windows 1.0 was released in 1985.
- In 1988 was published the first POSIX standard, in an effort to standardize compatibility between operating systems, at least for those akin to Unix.
- Linux was first released in 1991. Later on it was named GNU/Linux.

Operating Systems and Standards (2/4)

- Mac OS X was released in 2001, which implements most of the POSIX standard.
- The main detail that matters for OpenFOAM: an open-source CFD toolbox should rely on open-source technology and open standards.
- The detail that matters to a lot of users:
  - *Can I use it on Windows or Mac OS X?*
- What matters for making OpenFOAM work on most closed source OS':
  - *How to adapt the POSIX standard that is followed in OpenFOAM, to the systems we need it working on.*

Operating Systems and Standards (3/4)

- The result were a few unofficial variants of OpenFOAM:
  - For Windows, where POSIX is not supported, which requires a considerable effort in adapting the source code, depending on the approach.
  - For Mac OS X, which requires some effort in adapting the source code, since Mac OS X adopts most of the POSIX standard.
- Among these efforts, blueCFD was created in 2009, to improve upon existing work of porting OpenFOAM for Windows.
- In November 2013, blueCFD was rebranded to blueCFD®-Core, as our product line expanded.

Operating Systems and Standards (4/4)

- On the other hand, started on January 2016, official installation packages of OpenFOAM begun to appear that rely on Docker, a container strategy for providing easy to install virtual instances of Linux machines within a containment management software, namely Docker. Available packages:
  - OpenFOAM Foundation: openfoam.org/download/4-1-macos/
  - OpenCFD/ESI: openfoam.com/download/install-binary-windows.php
- Another container-like implementation is the *Windows Subsystem for Linux* in Windows 10, which allows using Ubuntu within it. Instructions:
  - OpenFOAM Foundation: openfoam.org/download/windows-10/
  - OpenCFD/ESI: openfoam.com/download/install-windows-10.php

What is blueCFD®-Core? (1/3)

- An open source project that provides high quality builds of OpenFOAM® for up-to-date Windows 7 to 10 64-bit, fully compilable on Windows.
- Complete functionality with the original scripts of OpenFOAM on Windows, by relying on MSys2.
- All features in OpenFOAM 4.x that require compiling, will build as intended in blueCFD-Core 2016.
- Customized solvers and libraries can also be compiled directly with OpenFOAM 4.x on Windows.
- Third-Party software is also provided, including: ParaView, Gnuplot, GDB, Notepad2, Meld, Python, etc…

What is blueCFD®-Core? (2/3)

- A Portable functionality, that allows copying the installed blueCFD-Core into an USB drive and ready to be used in other Windows machines.
- A single User Guide that addresses all major features of blueCFD-Core.
- Provide the full source code of OpenFOAM (including Git history for easy syncing and update), including the modifications done for making it work on Windows.

References:

- http://bluecfd.github.io/Core/
- http://bluecfd.github.io/Core/ReleaseNotes/

optimises your technology

Watermatex 2019

the international water association

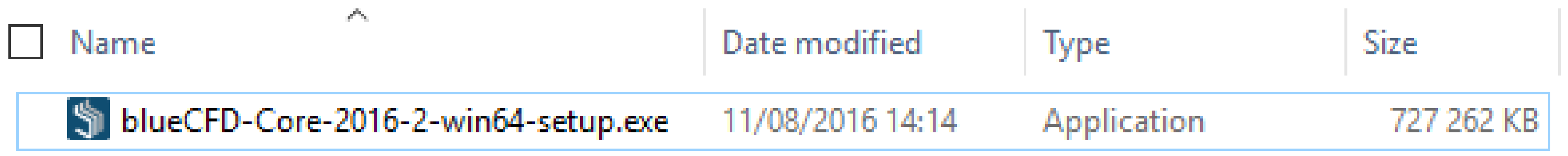
What is blueCFD®-Core? (3/3)

Objectives:

- Bring OpenFOAM technology to Windows, enabling all features available in GNU/Linux Distributions.
- Preserving full compatibility and functionality with the original source code, with the minimal impact to the source code.
- Quality assurance tests, in order to ensure and document which features are working in accordance with the official Linux distribution.
- Consolidating community efforts into a single project that ports OpenFOAM for native execution on Windows.

Installing blueCFD-Core (1/15)

In the provided USB should be the following file and folder:



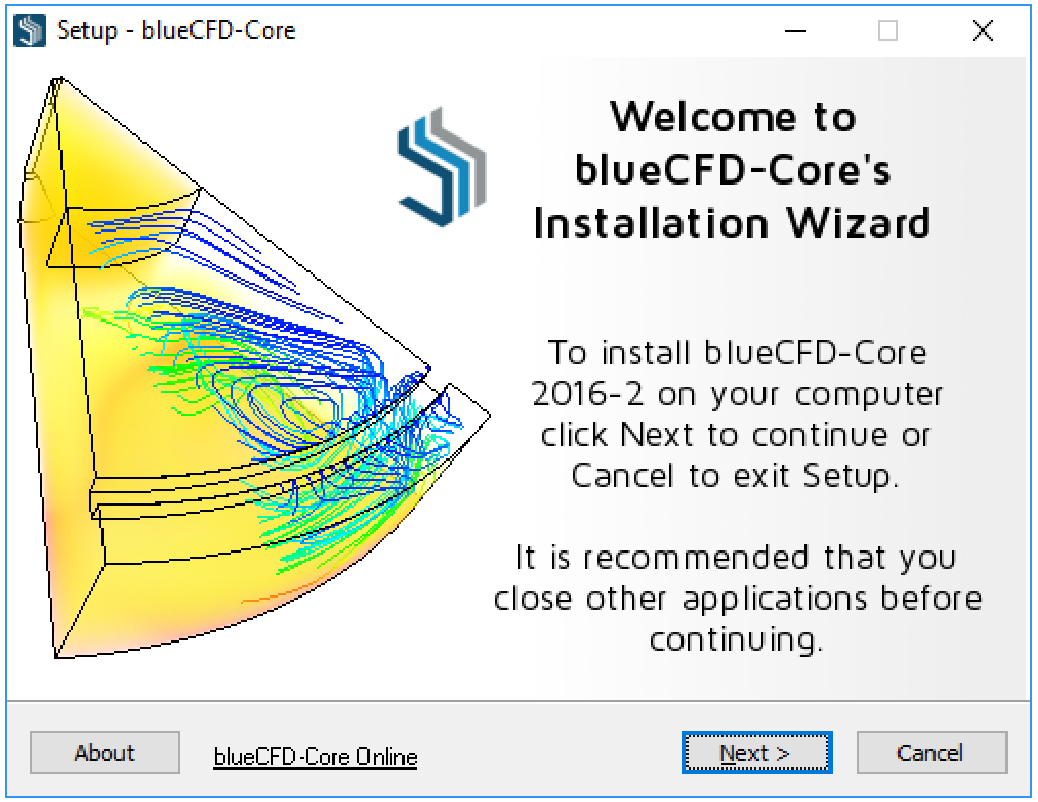
Name	Date modified	Type	Size
blueCFD-Core-2016-2-win64-setup.exe	11/08/2016 14:14	Application	727 262 KB

To start the installer, double click on the file
`blueCFD-Core-2016-2-win64-setup.exe`

It's also available online at http://bluecfd.github.io/Core/Downloads/

Installing blueCFD-Core (2/15)

Once the installer starts, it will show the following window:



Setup - blueCFD-Core

Welcome to blueCFD-Core's Installation Wizard

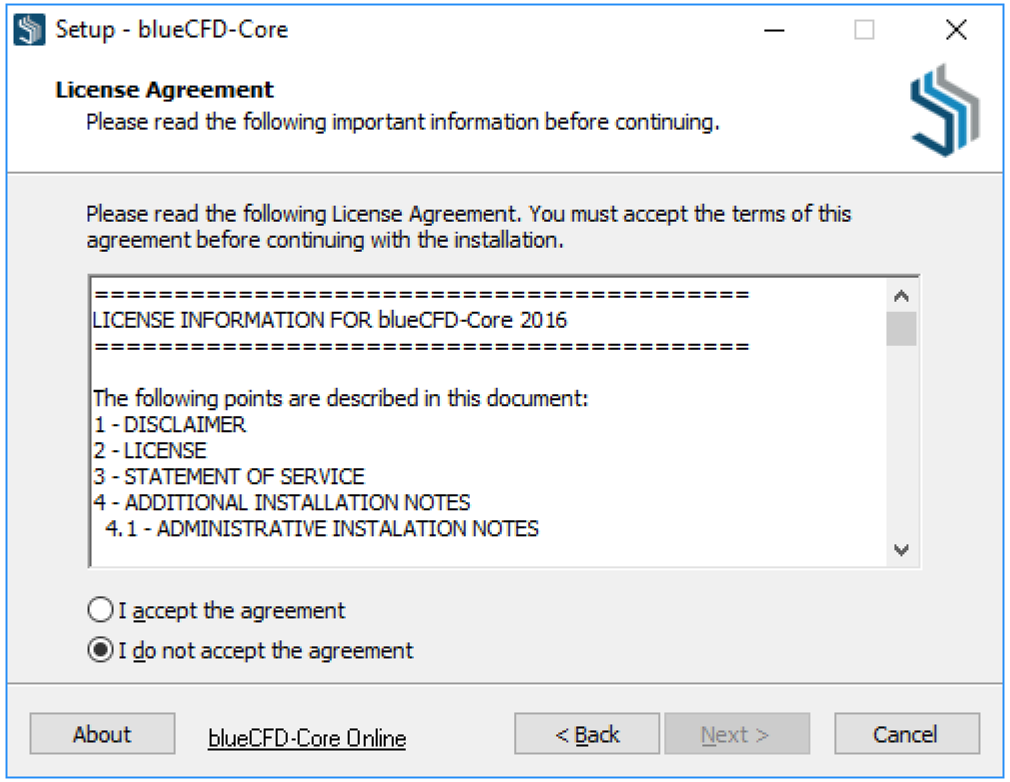
To install blueCFD-Core 2016-2 on your computer click Next to continue or Cancel to exit Setup.

It is recommended that you close other applications before continuing.

Click on the "Next" button.

Installing blueCFD-Core (3/15)

The next window provides the license information and the request for agreement:



Setup - blueCFD-Core

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

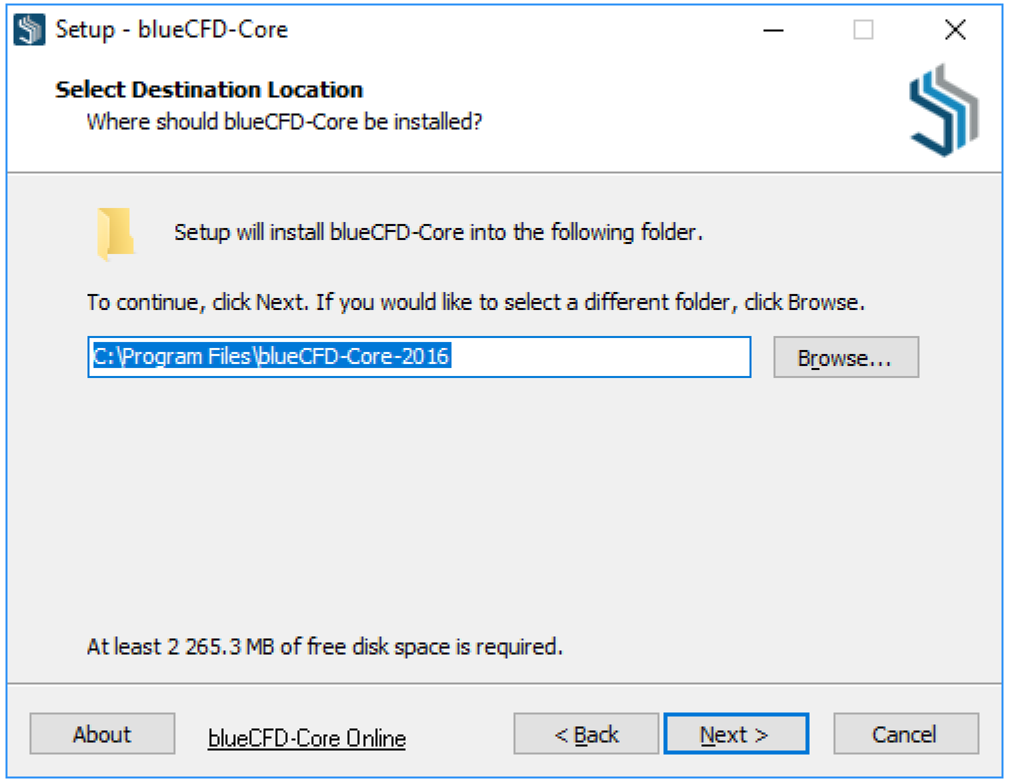
```
==========================================
LICENSE INFORMATION FOR blueCFD-Core 2016
==========================================

The following points are described in this document:
1 - DISCLAIMER
2 - LICENSE
3 - STATEMENT OF SERVICE
4 - ADDITIONAL INSTALLATION NOTES
  4.1 - ADMINISTRATIVE INSTALATION NOTES
```

( ) I accept the agreement
(●) I do not accept the agreement

After accepting the agreement, click on the “Next” button.

Installing blueCFD-Core (4/15)

In the next window, it asks where blueCFD-Core should be installed:



Setup - blueCFD-Core

Select Destination Location

Where should blueCFD-Core be installed?

Setup will install blueCFD-Core into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\blueCFD-Core-2016

At least 2 265.3 MB of free disk space is required.

Notes in the next slide...

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Installing blueCFD-Core (5/15)

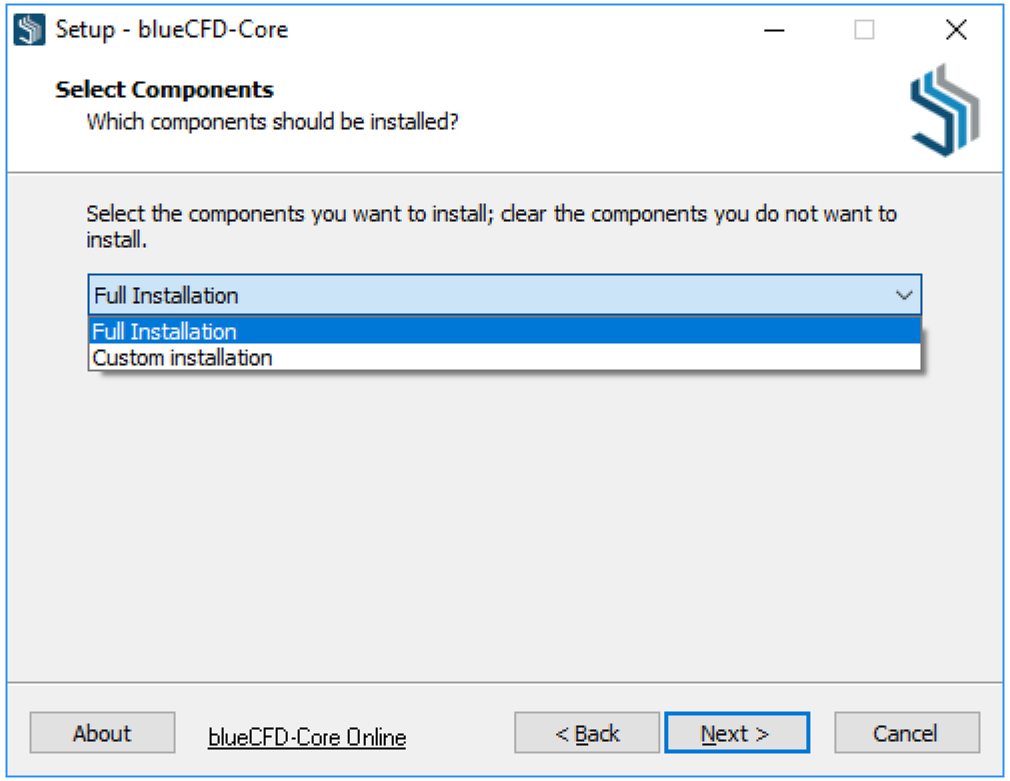
Notes on “Select Destination Location”:

- The *standard location should work for most people*, although keep in mind that the installer will activate the ability to write files within specific user sub-folders inside this folder.
- Alternatively, you can install in “`C:\blueCFD-Core-2016`” or in a similar drive letter.
- Or if you prefer, you can install this only for your own personal area, by closing the installer and running it manually from the command line, like this:

```
blueCFD-Core-2016-2-win64-setup.exe /SINGLEUSER=1
```

Installing blueCFD-Core (6/15)

Once the location is chosen, click on the “Next” button and it will ask what type of installation to perform:


Setup - blueCFD-Core

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install.

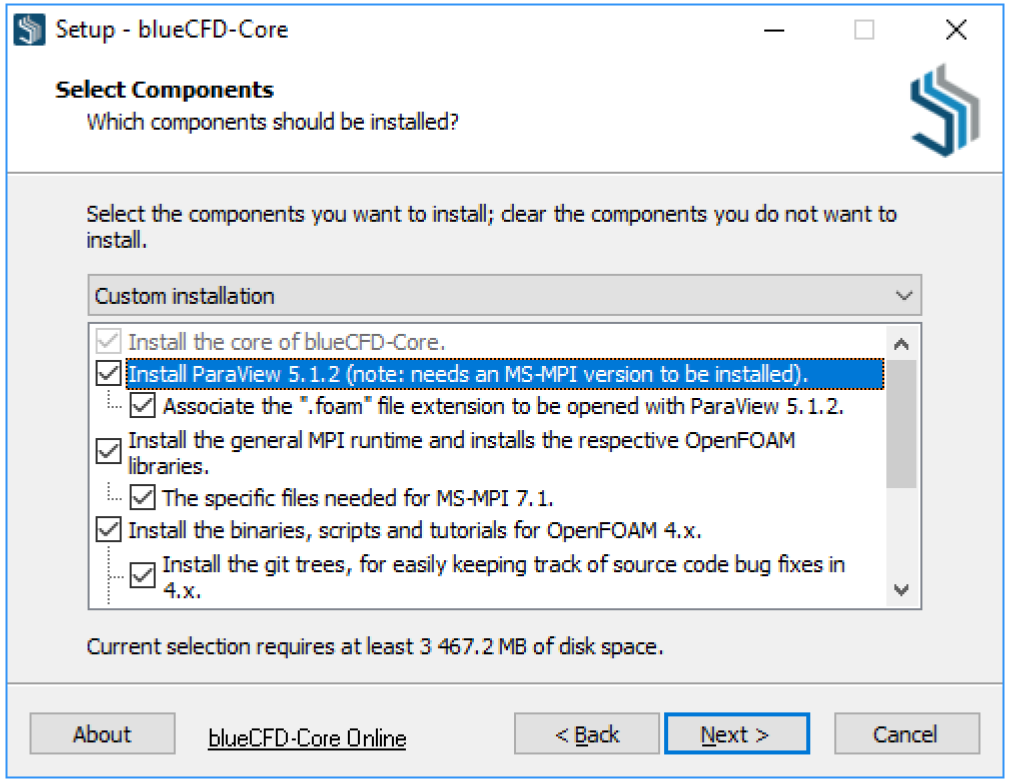
Full Installation

More details on the next slide…

Installing blueCFD-Core (7/15)

The types of installation are essentially:

- "Full installation" – For installing everything.
- "Custom installation" – For choosing which features to install, for example:



Setup - blueCFD-Core

Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install.

Custom installation

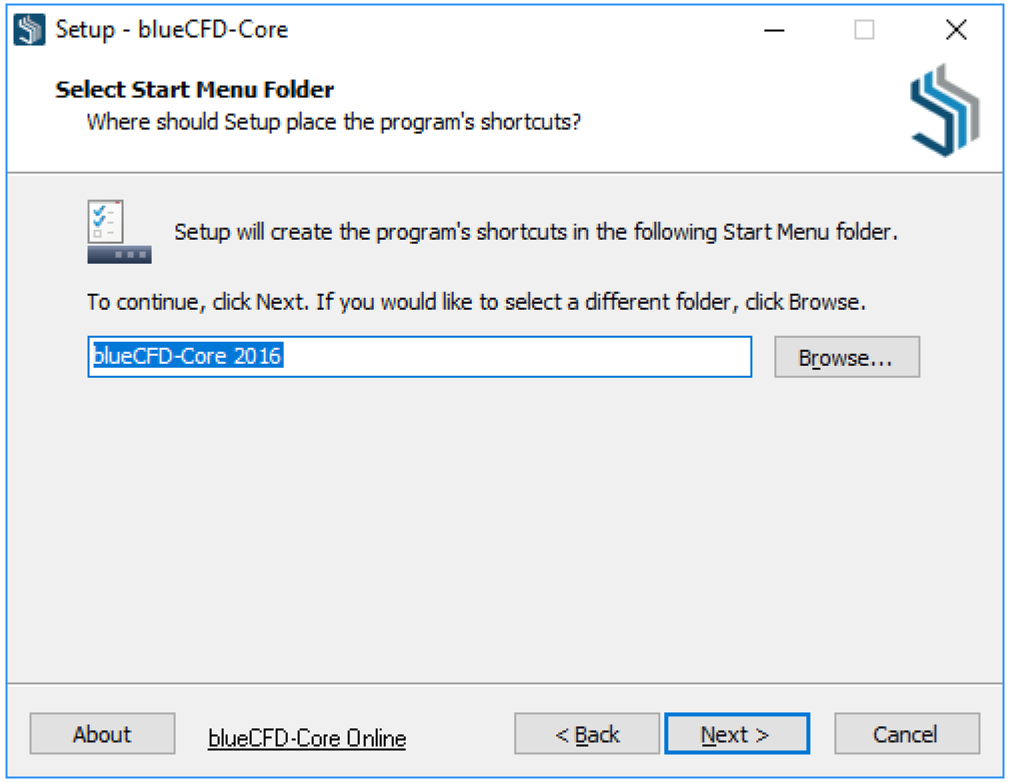
- [✓] Install the core of blueCFD-Core.
- [✓] Install ParaView 5.1.2 (note: needs an MS-MPI version to be installed).
  - [✓] Associate the ".foam" file extension to be opened with ParaView 5.1.2.
- [✓] Install the general MPI runtime and installs the respective OpenFOAM libraries.
  - [✓] The specific files needed for MS-MPI 7.1.
- [✓] Install the binaries, scripts and tutorials for OpenFOAM 4.x.
  - [✓] Install the git trees, for easily keeping track of source code bug fixes in 4.x.

Current selection requires at least 3 467.2 MB of disk space.

FS DYNAMICS optimises your technology

Installing blueCFD-Core (8/15)

Once the choices have been made, click on the “Next” button, which will allow choosing the *Start Menu* group where the blueCFD-Core shortcuts should be placed:


Setup - blueCFD-Core

Select Start Menu Folder

Where should Setup place the program's shortcuts?

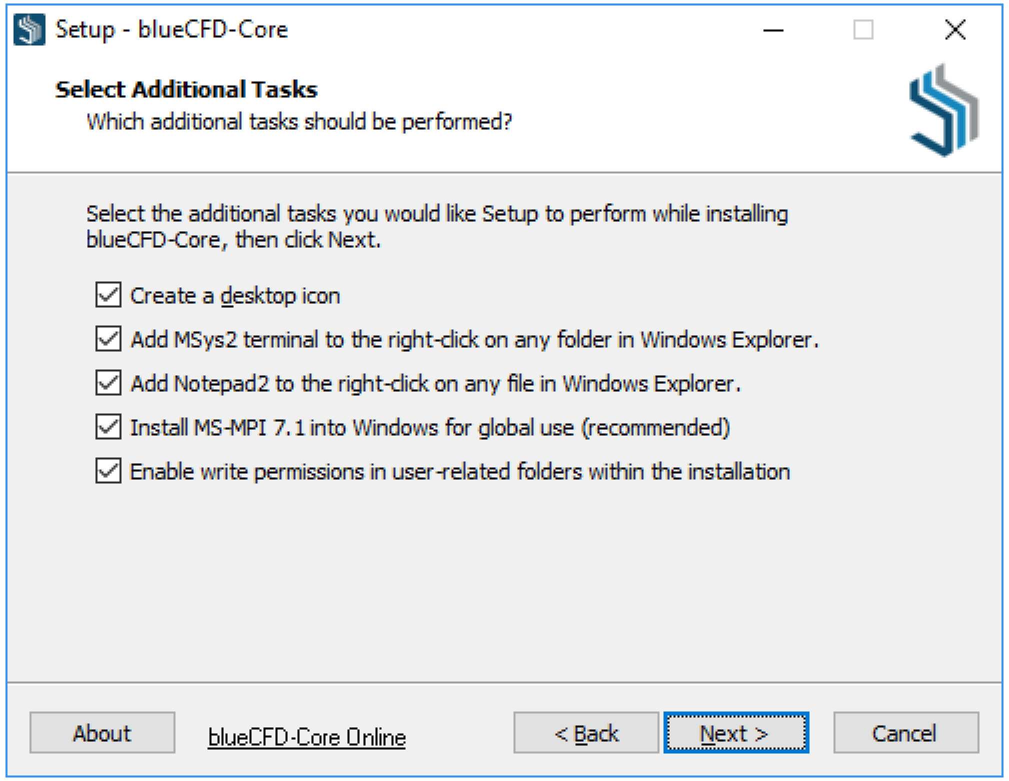
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

blueCFD-Core 2016

Installing blueCFD-Core (9/15)

After choosing the group name, click on the "Next" button, which lead to the window with the following options:


Setup - blueCFD-Core

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing blueCFD-Core, then click Next.

- [✓] Create a desktop icon
- [✓] Add MSys2 terminal to the right-click on any folder in Windows Explorer.
- [✓] Add Notepad2 to the right-click on any file in Windows Explorer.
- [✓] Install MS-MPI 7.1 into Windows for global use (recommended)
- [✓] Enable write permissions in user-related folders within the installation

Details in the next slide…

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Installing blueCFD-Core (10/15)

Notes regarding “Select Additional Tasks” (1/2):

- The desktop icon is useful specially on Windows 8, 8.1 and 10, due to either the non-existence of a Start Menu (Windows 8) or because the sub-folders are not longer displayed (Windows 10).
  - Without these icons, it could get very complicated to use blueCFD-Core on those versions of Windows.
- The option to "Add Notepad2 to the right-click on any file in Windows Explorer" is useful for editing the OpenFOAM case files.
- This option to “Add MSys2 terminal to the right-click on any folder in Windows Explorer” is also very useful.

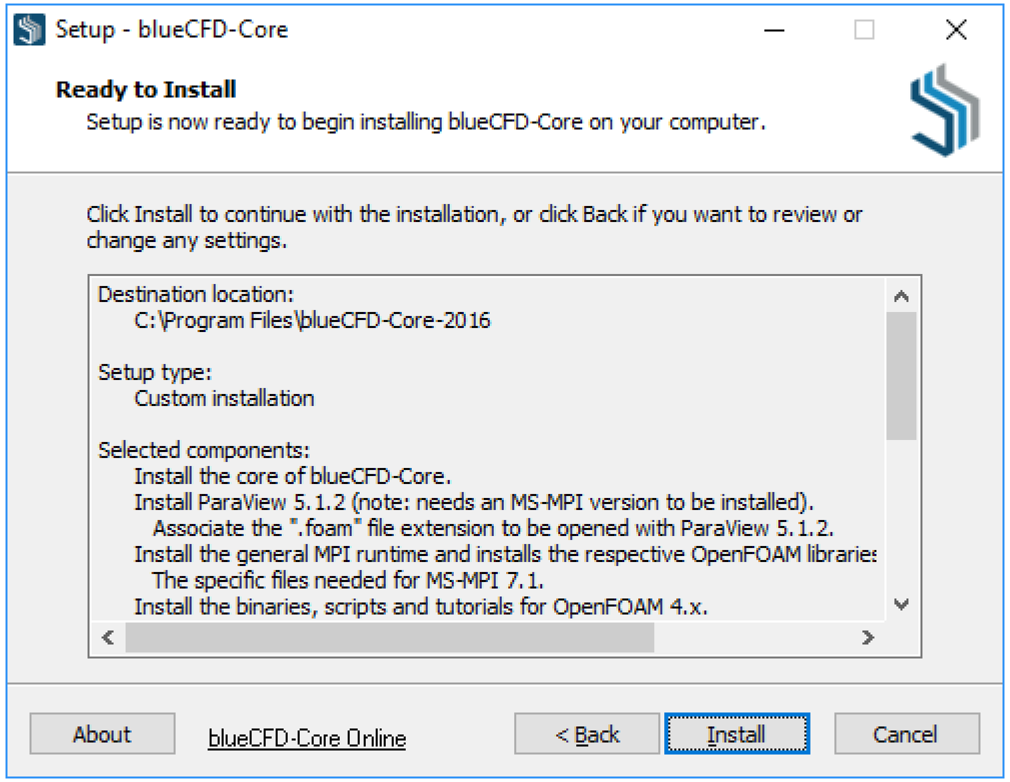
Installing blueCFD-Core (10/15)

Notes regarding “Select Additional Tasks” (2/2):

- The option to “Install MS-MPI 7.1 for global use” will install MS-MPI directly into Windows’ system folders.
- The option to "enable write permissions" is necessary and advisable when the user currently installing blueCFD-Core is able to perform administrative installations.
  - Needed when installing in the default folder: `C:\Program Files`
  - If not enabled in this situation, namely to give the ability to write in the main user folders “`ofuser-4.x`”, “`msys64\home\ofuser`” and “`msys64\etc`”, will disrupt the conventional installation process.

Installing blueCFD-Core (11/15)

Once the choices have been made, click on the "Next" button. The final window before the installation begins is shown:



Ready to Install

Setup is now ready to begin installing blueCFD-Core on your computer.

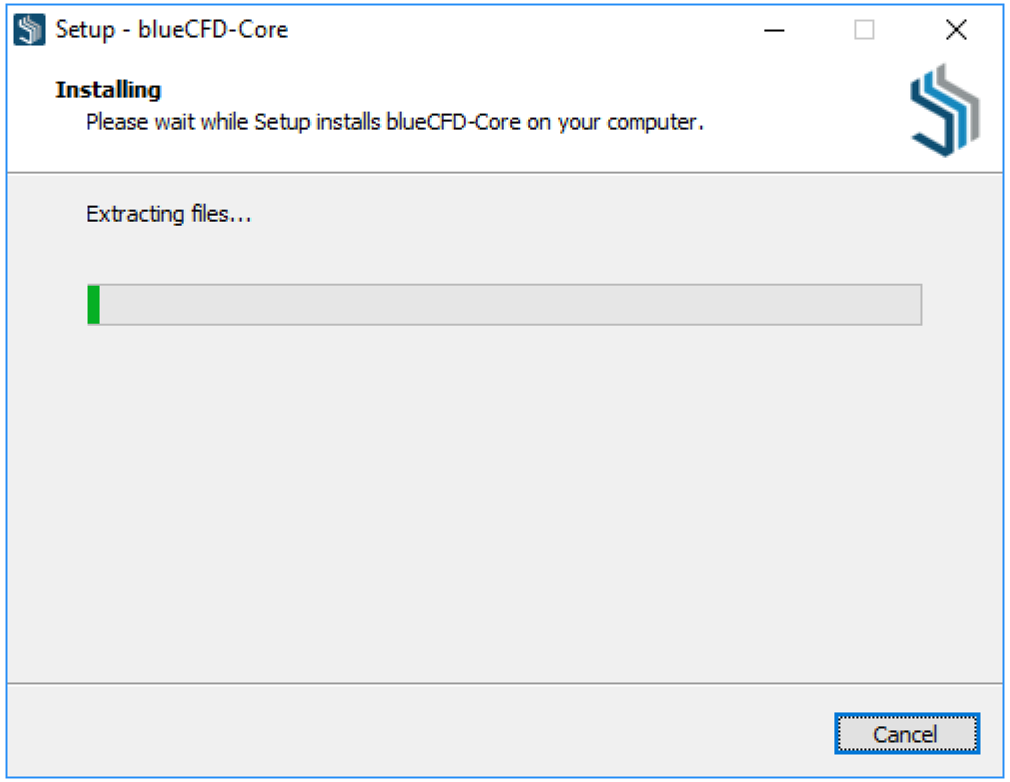
Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
    C:\Program Files\blueCFD-Core-2016

Setup type:
    Custom installation

Selected components:
    Install the core of blueCFD-Core.
    Install ParaView 5.1.2 (note: needs an MS-MPI version to be installed).
      Associate the ".foam" file extension to be opened with ParaView 5.1.2.
    Install the general MPI runtime and installs the respective OpenFOAM librarie
      The specific files needed for MS-MPI 7.1.
    Install the binaries, scripts and tutorials for OpenFOAM 4.x.
```

Click on the "Install" button to proceed.

Installing blueCFD-Core (12/15)

While it is installing blueCFD-Core, it should show the progress bar, as exemplified here:



Setup - blueCFD-Core

Installing

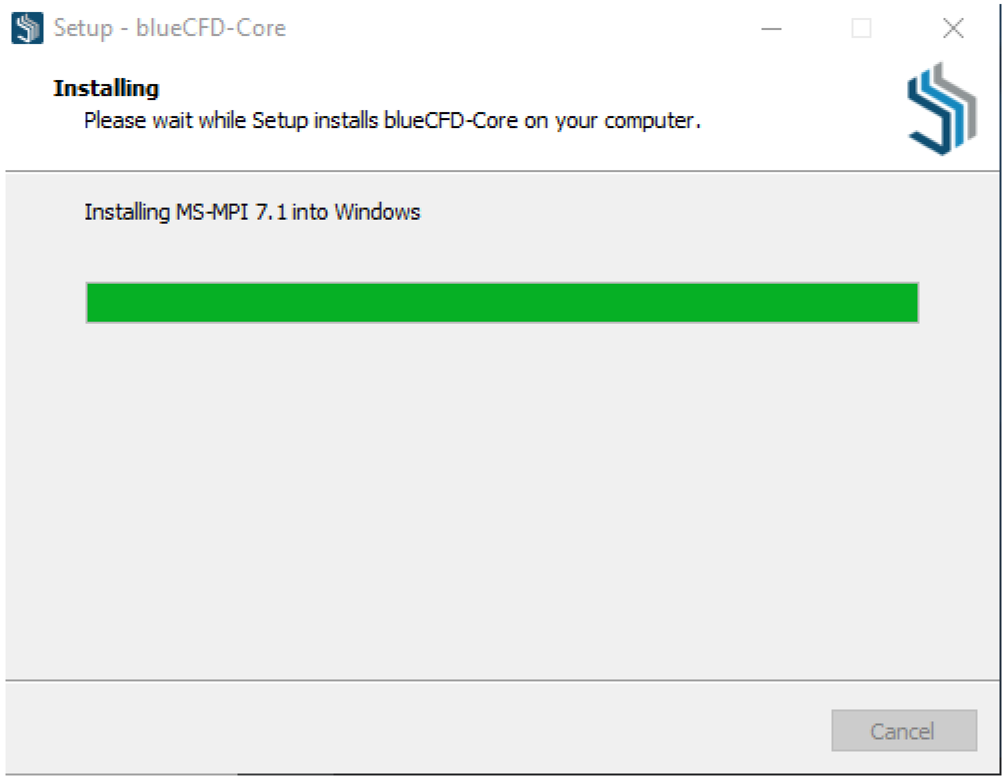
Please wait while Setup installs blueCFD-Core on your computer.

Extracting files...

The progress bar will go forward …

Installing blueCFD-Core (13/15)

When it reaches the end of the files to be installed, it will run the external installers (MS-MPI), if selected. This will reset the progress bar for this second progress stage:


Setup - blueCFD-Core

Installing

Please wait while Setup installs blueCFD-Core on your computer.

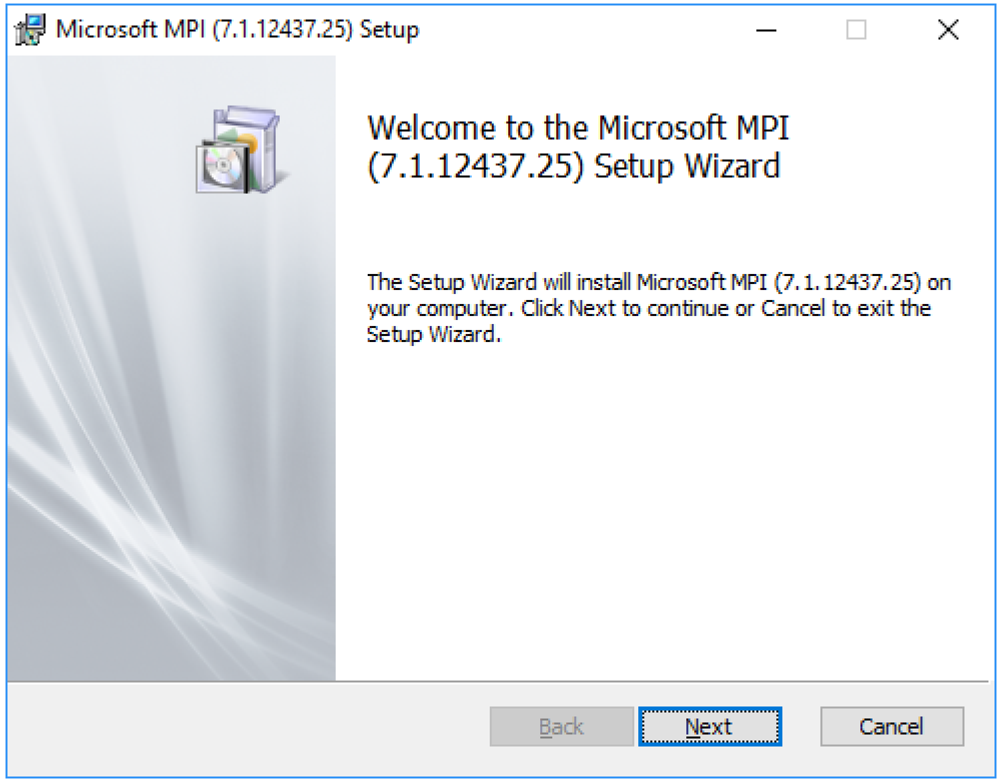
Installing MS-MPI 7.1 into Windows

Installing blueCFD-Core (14/15)

One of the possible steps in this second progress stage is to install MS-MPI, which will interactively ask you to follow its own installation steps.

The steps should be fairly simple:

1. Introduction.
2. Accept the license.
3. Select location.
4. Click on the “Install” button.
5. Wait a little while.
6. Click on the “Finish” button.



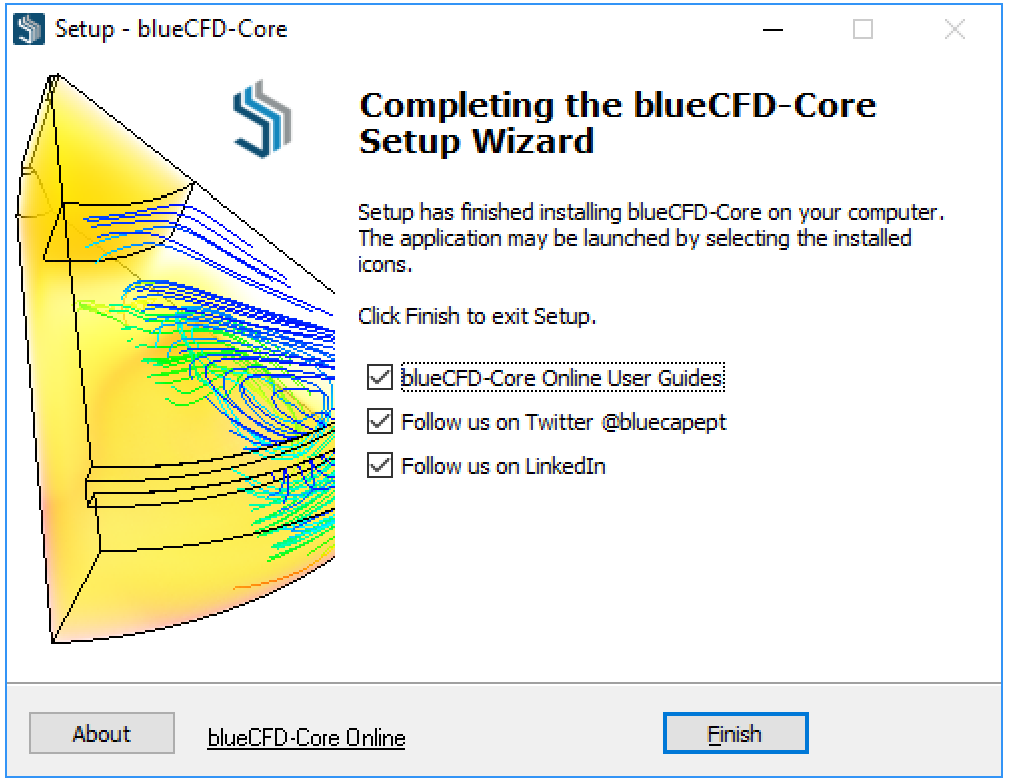
Welcome to the Microsoft MPI (7.1.12437.25) Setup Wizard

The Setup Wizard will install Microsoft MPI (7.1.12437.25) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

The control will then return to the blueCFD-Core installer.

Installing blueCFD-Core (15/15)

Once the installation is complete, it will show the following window:


Setup - blueCFD-Core

Completing the blueCFD-Core Setup Wizard

Setup has finished installing blueCFD-Core on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

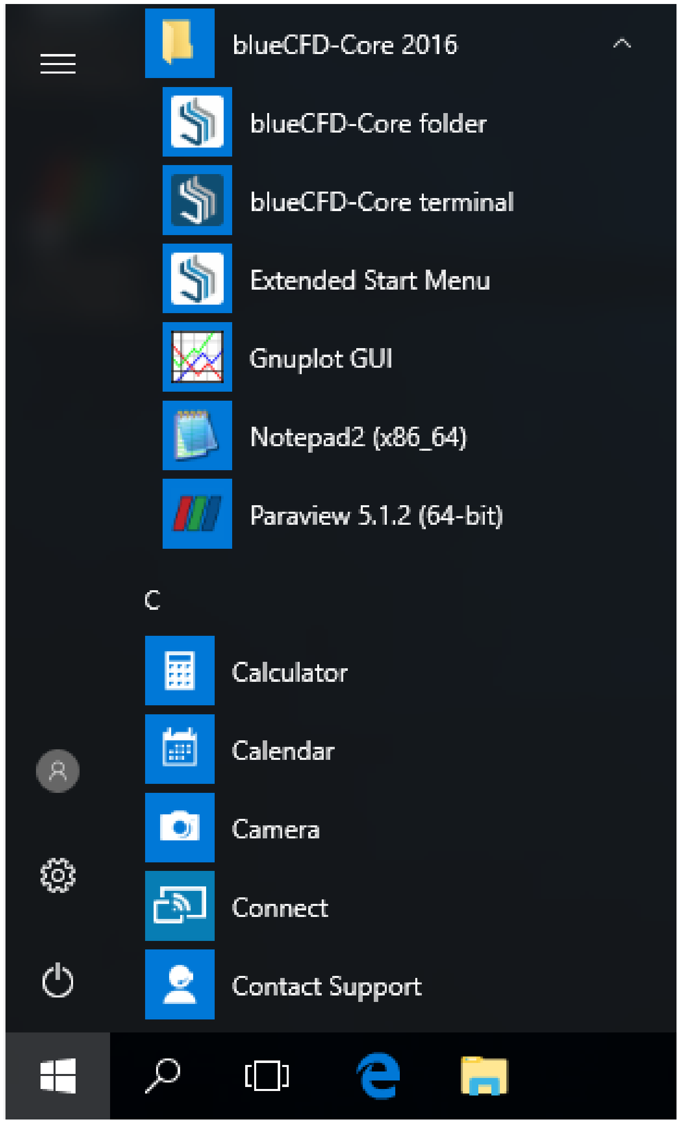
- [✓] blueCFD-Core Online User Guides
- [✓] Follow us on Twitter @bluecapept
- [✓] Follow us on LinkedIn

Once you click in the "Finish" button, blueCFD-Core should be installed with the chosen features!

Overview of installed packages (1/10)

The Start Menu, as shown on the right for Windows 10, is the conventional way to access installed applications in Windows. The structure depends on each Windows version:

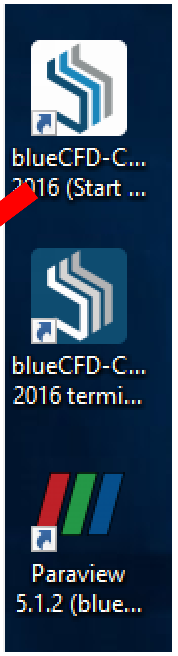
- On Windows 7, click on “All Programs” and scroll down for blueCFD-Core-2016, which provides a complete tree with branches.
- For Windows 8 to 10, it’s easier to use the desktop icons as seen in the next slide, but as shown to the right, it only shows a summary branch that lists all shortcuts.

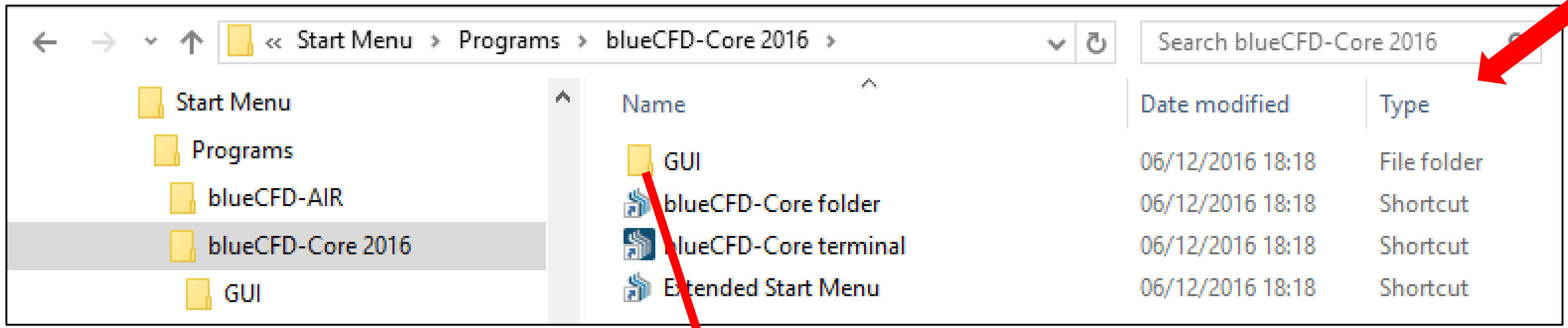


Overview of installed packages (2/10)

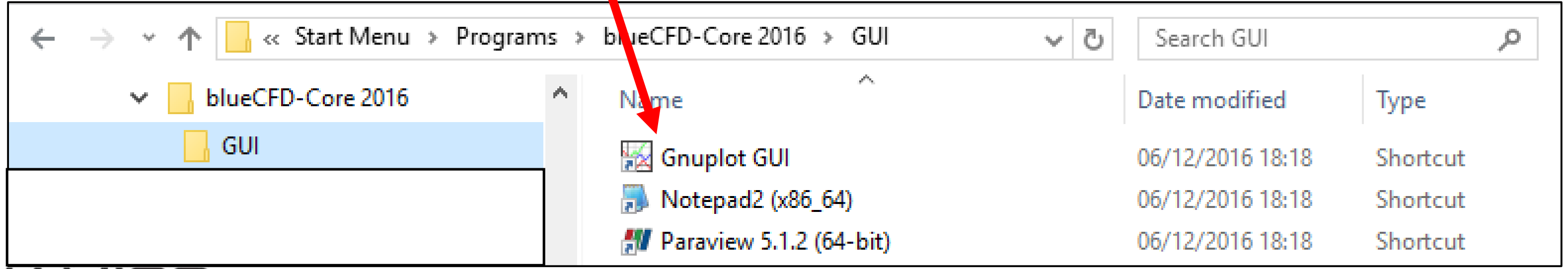
On Windows 8 to 10, blueCFD-Core provides a few shortcuts in your Windows Desktop, as shown on the right.

The first icon is the “blueCFD-Core 2016 (Start Menu)”, that provides access to the contents that were shown on the previous slide.





Name	Date modified	Type
GUI	06/12/2016 18:18	File folder
blueCFD-Core folder	06/12/2016 18:18	Shortcut
blueCFD-Core terminal	06/12/2016 18:18	Shortcut
Extended Start Menu	06/12/2016 18:18	Shortcut



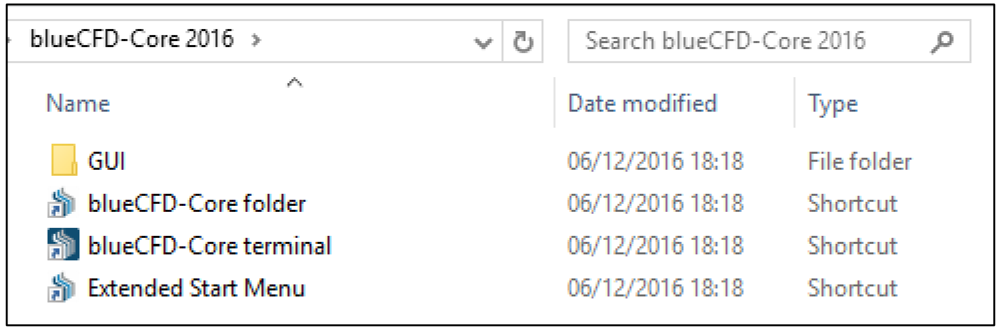
Name	Date modified	Type
Gnuplot GUI	06/12/2016 18:18	Shortcut
Notepad2 (x86_64)	06/12/2016 18:18	Shortcut
Paraview 5.1.2 (64-bit)	06/12/2016 18:18	Shortcut

FS DYNAMICS optimises your technology

Overview of installed packages (3/10)

Additional Shortcuts:

- **blueCFD-Core folder** – Shortcut that leads to the folder where blueCFD-Core is installed.



blueCFD-Core 2016 ›

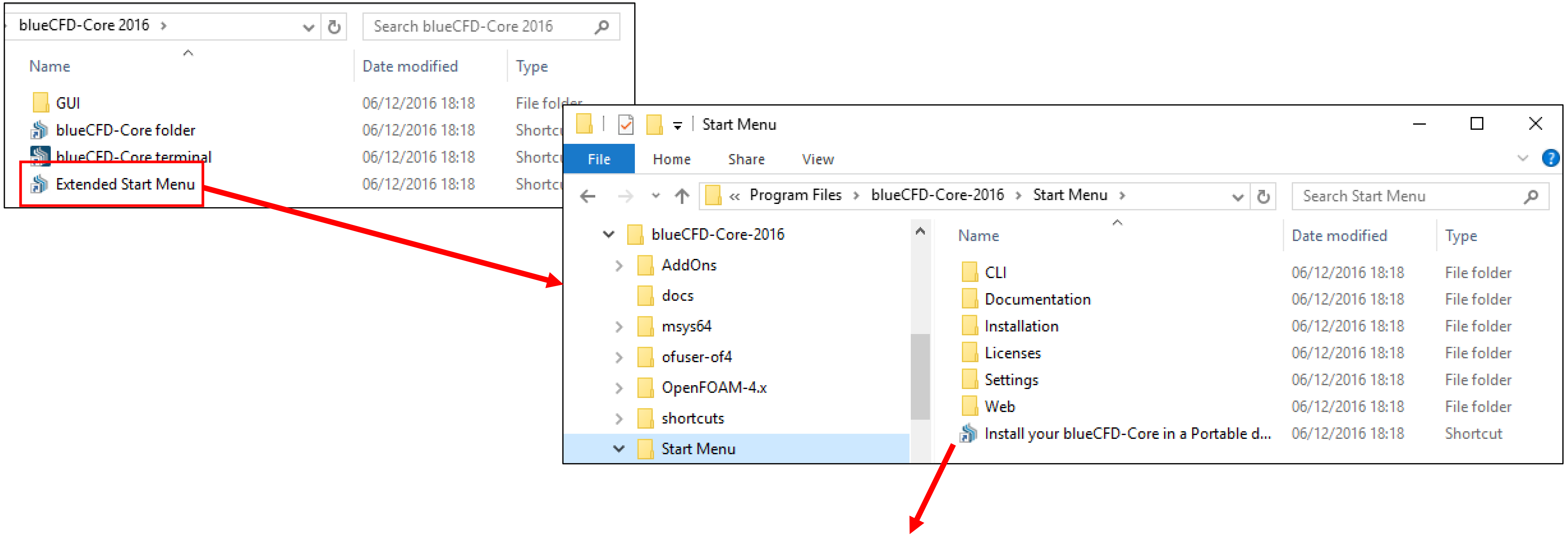
Search blueCFD-Core 2016

Name	Date modified	Type
GUI	06/12/2016 18:18	File folder
blueCFD-Core folder	06/12/2016 18:18	Shortcut
blueCFD-Core terminal	06/12/2016 18:18	Shortcut
Extended Start Menu	06/12/2016 18:18	Shortcut

- **blueCFD-Core terminal** – Primary command based interface for OpenFOAM, similar to how it works on Linux (using MSys2).
- **Extended Start Menu** – will go into the blueCFD-Core's installation folder to access additional documentation and low-level functionality.

Overview of installed packages (4/10)

Double-clicking on “Extended Start Menu”, will go into:



blueCFD-Core 2016 › Search blueCFD-Core 2016

Name	Date modified	Type
GUI	06/12/2016 18:18	File folder
blueCFD-Core folder	06/12/2016 18:18	Shortcu
blueCFD-Core terminal	06/12/2016 18:18	Shortcu
Extended Start Menu	06/12/2016 18:18	Shortcu

Start Menu

File Home Share View

« Program Files › blueCFD-Core-2016 › Start Menu › Search Start Menu

blueCFD-Core-2016
- AddOns
- docs
- msys64
- ofuser-of4
- OpenFOAM-4.x
- shortcuts
- Start Menu

Name	Date modified	Type
CLI	06/12/2016 18:18	File folder
Documentation	06/12/2016 18:18	File folder
Installation	06/12/2016 18:18	File folder
Licenses	06/12/2016 18:18	File folder
Settings	06/12/2016 18:18	File folder
Web	06/12/2016 18:18	File folder
Install your blueCFD-Core in a Portable d...	06/12/2016 18:18	Shortcut

This application copies the whole blueCFD-Core installation to a portable drive.

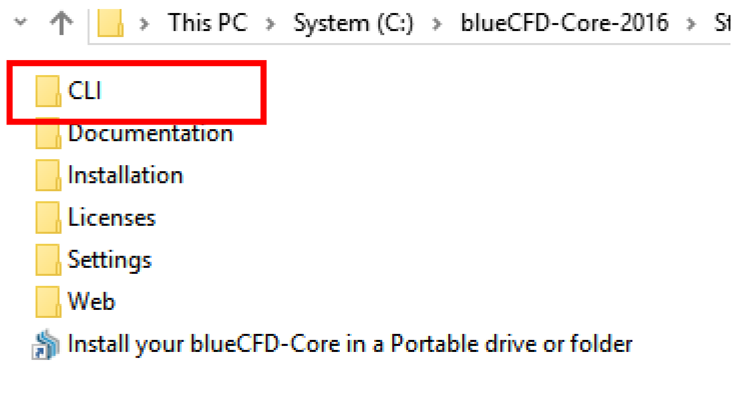
FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Overview of installed packages (5/10)

CLI (Command Line Interface) folder:

- **Gnuplot Shell** – text-based command line interaction for working with Gnuplot.
- **Python 2 Shell (x86_64)** – Shortcut to a CLI for Python 2.7 (provided with MSys2).
- **Python 3 Shell (x86_64)** – Shortcut to a CLI for Python 3.5 (provided with MSys2).



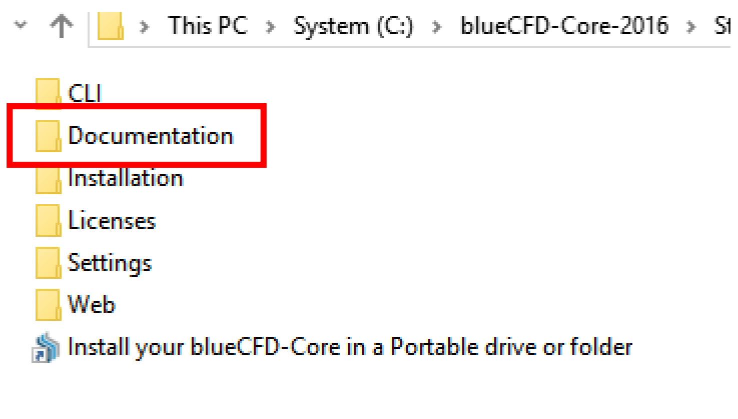
This PC › System (C:) › blueCFD-Core-2016 › St
CLI
Documentation
Installation
Licenses
Settings
Web
Install your blueCFD-Core in a Portable drive or folder

Note: These do not provide access to OpenFOAM on their own, i.e. additional coding is needed.

Overview of installed packages (6/10)

Documentation folder:

- **blueCFD-Core Online User Guides** – Link to online User Guides page.



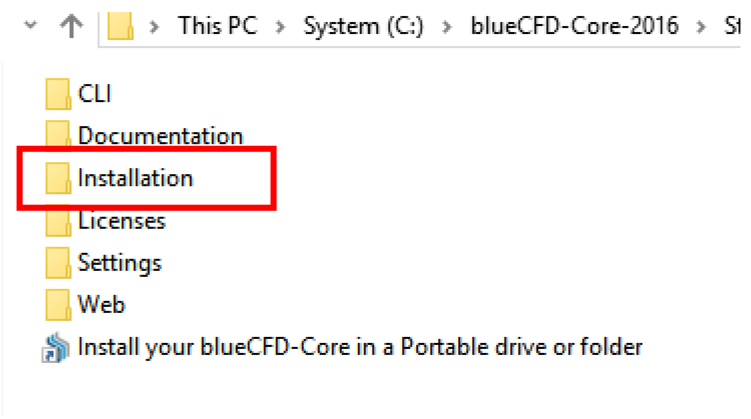
This PC › System (C:) › blueCFD-Core-2016 › St

CLI
Documentation
Installation
Licenses
Settings
Web
Install your blueCFD-Core in a Portable drive or folder

- **blueCFD-Core Release Notes** – Link to online page.
- **OpenFOAM User Guide** – One of the most important documents for learning how to use OpenFOAM.
- **Gnuplot** – Folder with links and shortcuts for Gnuplot's documentation.
- **ParaView Guides** – Folder with links and shortcuts for ParaView's documentation.

FS DYNAMICS optimises your technology

Overview of installed packages (8/10)

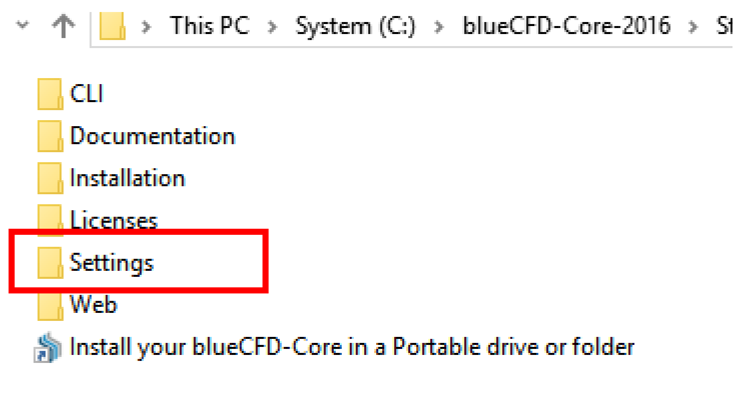

This PC › System (C:) › blueCFD-Core-2016 › St
CLI
Documentation
Installation
Licenses
Settings
Web
Install your blueCFD-Core in a Portable drive or folder

Installation folder:

- **System-wide install of MS-MPI 7.1** – MS-MPI installer's shortcut. Useful when not chosen by default during blueCFD-Core's installation.
- **Uninstall blueCFD-Core 2016** – Uninstaller application link to remove blueCFD-Core 2016.
  - Note: The uninstaller will not delete any files or folders that have been created during the normal use of blueCFD-Core.

Overview of installed packages (9/10)

Settings folder:

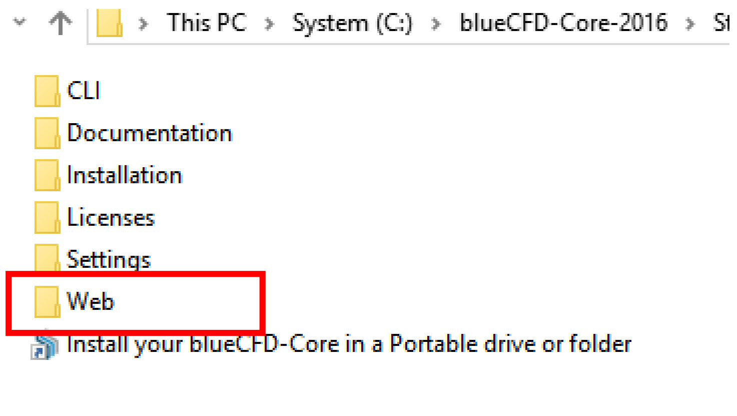

This PC › System (C:) › blueCFD-Core-2016 › S
CLI
Documentation
Installation
Licenses
Settings
Web
Install your blueCFD-Core in a Portable drive or folder

- **Local Drive Mode** – Shortcuts for activating and deactivating a virtual drive to the blueCFD-Core installation folder.
    - useful when re-building OpenFOAM from source code or building custom source code.
    - See wiki pages for more details: github.com/blueCFD/Core/wiki/
- **MPI mode** – Not entirely useful in blueCFD-Core 2016, but designed to allow changing between MPI toolboxes, depending on those installed on your system.
    - Contact us if you need another version, email address is on the presentation cover.

Overview of installed packages (10/10)

Web folder:

- Links to online websites which provide information for the software provided with blueCFD-Core. This is where most of the remaining documentation can be found.

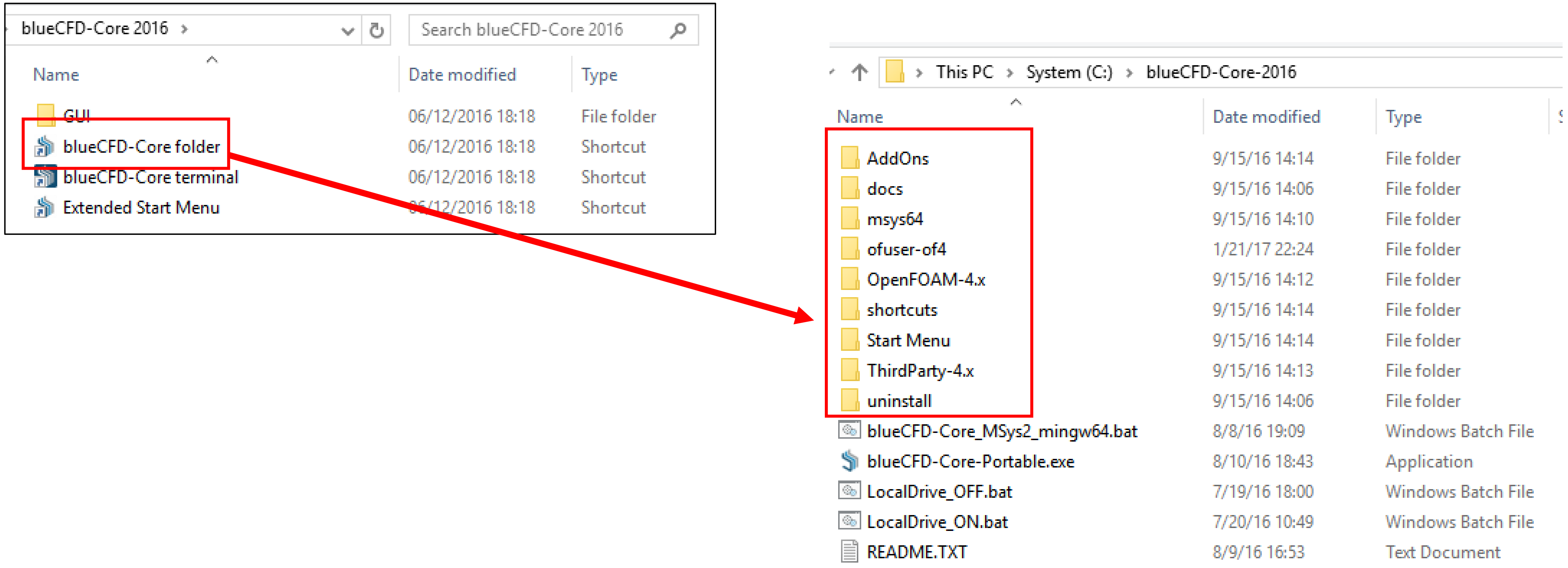

This PC › System (C:) › blueCFD-Core-2016 › St

CLI
Documentation
Installation
Licenses
Settings
Web
Install your blueCFD-Core in a Portable drive or folder

FS DYNAMICS optimises your technology

Overview of installation directory (1/3)

Double-clicking on "blueCFD-Core folder", will go into:



blueCFD-Core 2016 ›

Name	Date modified	Type
GUI	06/12/2016 18:18	File folder
blueCFD-Core folder	06/12/2016 18:18	Shortcut
blueCFD-Core terminal	06/12/2016 18:18	Shortcut
Extended Start Menu	06/12/2016 18:18	Shortcut

This PC › System (C:) › blueCFD-Core-2016

Name	Date modified	Type
AddOns	9/15/16 14:14	File folder
docs	9/15/16 14:06	File folder
msys64	9/15/16 14:10	File folder
ofuser-of4	1/21/17 22:24	File folder
OpenFOAM-4.x	9/15/16 14:12	File folder
shortcuts	9/15/16 14:14	File folder
Start Menu	9/15/16 14:14	File folder
ThirdParty-4.x	9/15/16 14:13	File folder
uninstall	9/15/16 14:06	File folder
blueCFD-Core_MSys2_mingw64.bat	8/8/16 19:09	Windows Batch File
blueCFD-Core-Portable.exe	8/10/16 18:43	Application
LocalDrive_OFF.bat	7/19/16 18:00	Windows Batch File
LocalDrive_ON.bat	7/20/16 10:49	Windows Batch File
README.TXT	8/9/16 16:53	Text Document

FS DYNAMICS optimises your technology

Overview of installation directory (2/3)

In the main installation folder, the most important folders are:

- **AddOns** – Additional software, such as ParaView, Gnuplot, etc…
- **msys64** – “Minimal System” (MSys2 64-bit), similar to a terminal interface in a Linux Distribution.
- **ofuser-of4** – Where your personal simulations and source code can be placed.
- **OpenFOAM-4.x** – OpenFOAM’s source code, binaries, tutorials and code documentation.
- **shortcuts** – Shortcuts for a portable installation.
- **Start Menu** – Shortcuts already described in the previous slides.
- **ThirdParty-4.x** – Third-party software that OpenFOAM needs but not provided by MSys2.

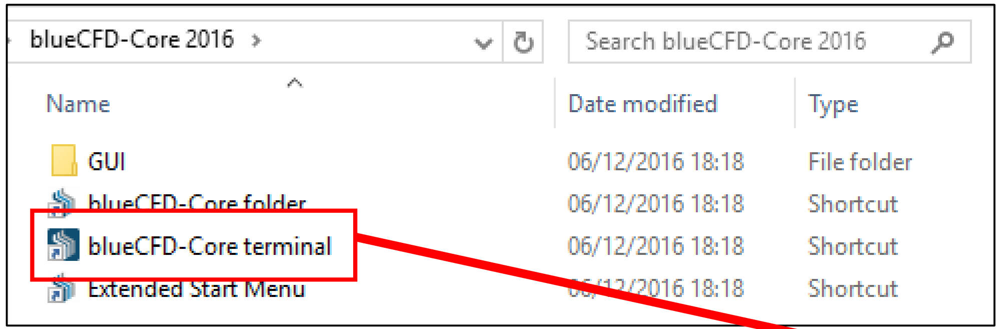
Overview of installation directory (3/3)

Important sub-folders:

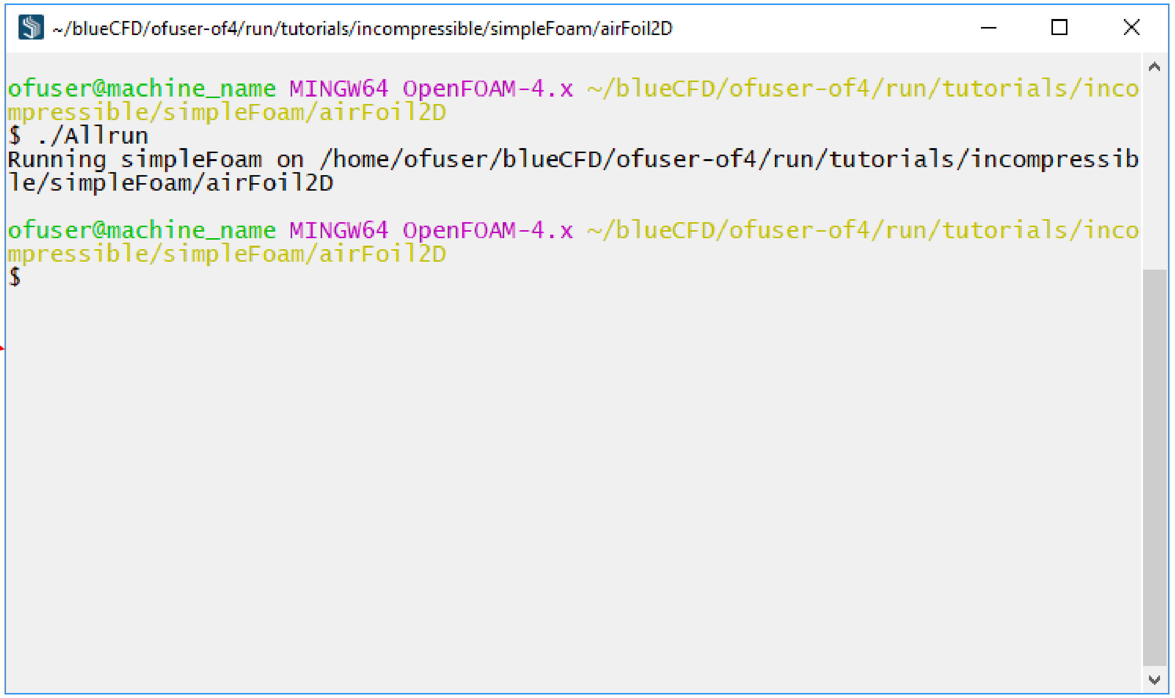
- **msys64\home\ofuser** – Where the Msys2 shell environment will start and where most personal files are stored.
- **ofuser-of4\run** – Where your personal simulations cases should be placed.
- **OpenFOAM-4.x:**
  - **doc** – Location for OpenFOAM's documentation.
  - **tutorials** – Location for the original copy of the OpenFOAM tutorial case folders.

Command Line Interface

The main interface available in blueCFD-Core is essentially the same that is available in OpenFOAM: the *Command Line Interface* (CLI).



```
blueCFD-Core 2016 >                          Search blueCFD-Core 2016

Name                          Date modified       Type
GUI                           06/12/2016 18:18    File folder
blueCFD-Core folder           06/12/2016 18:18    Shortcut
blueCFD-Core terminal         06/12/2016 18:18    Shortcut
Extended Start Menu           06/12/2016 18:18    Shortcut
```



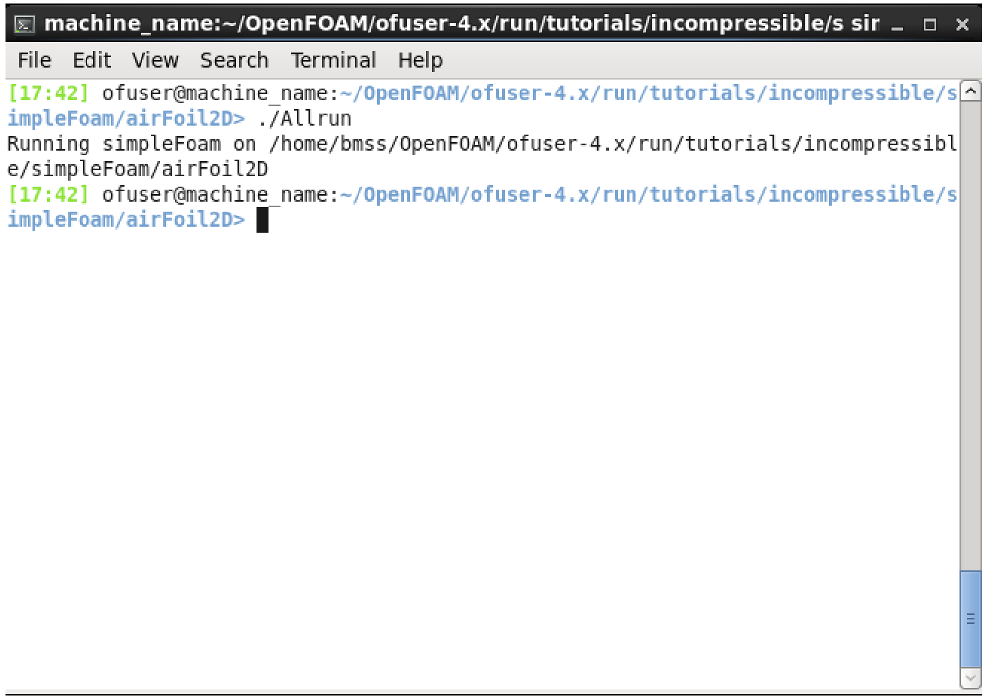
```
~/blueCFD/ofuser-of4/run/tutorials/incompressible/simpleFoam/airFoil2D

ofuser@machine_name MINGW64 OpenFOAM-4.x ~/blueCFD/ofuser-of4/run/tutorials/inco
mpressible/simpleFoam/airFoil2D
$ ./Allrun
Running simpleFoam on /home/ofuser/blueCFD/ofuser-of4/run/tutorials/incompressib
le/simpleFoam/airFoil2D

ofuser@machine_name MINGW64 OpenFOAM-4.x ~/blueCFD/ofuser-of4/run/tutorials/inco
mpressible/simpleFoam/airFoil2D
$
```

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

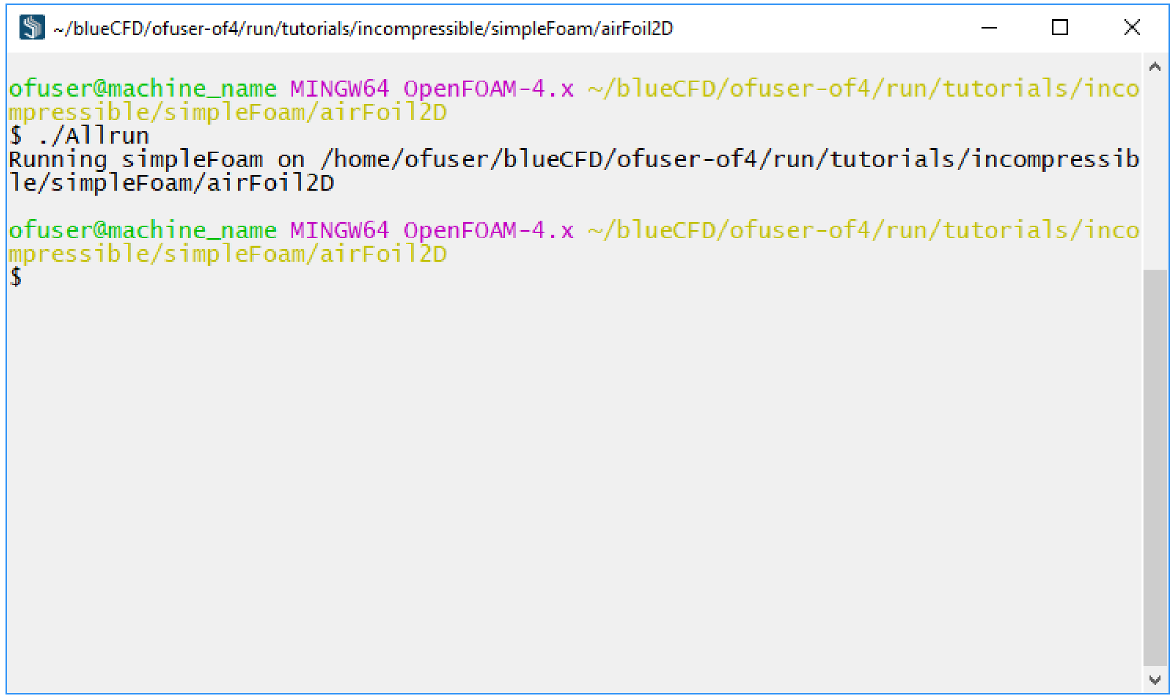
Getting started with the interface (1/12)

The main interface available in blueCFD-Core is essentially the same that is available in OpenFOAM: the *Command Line Interface* (CLI).


machine_name:~/OpenFOAM/ofuser-4.x/run/tutorials/incompressible/s sir
File Edit View Search Terminal Help

```
[17:42] ofuser@machine_name:~/OpenFOAM/ofuser-4.x/run/tutorials/incompressible/s
impleFoam/airFoil2D> ./Allrun
Running simpleFoam on /home/bmss/OpenFOAM/ofuser-4.x/run/tutorials/incompressibl
e/simpleFoam/airFoil2D
[17:42] ofuser@machine_name:~/OpenFOAM/ofuser-4.x/run/tutorials/incompressible/s
impleFoam/airFoil2D>
```

Interface on Linux


~/blueCFD/ofuser-of4/run/tutorials/incompressible/simpleFoam/airFoil2D

```
ofuser@machine_name MINGW64 OpenFOAM-4.x ~/blueCFD/ofuser-of4/run/tutorials/inco
mpressible/simpleFoam/airFoil2D
$ ./Allrun
Running simpleFoam on /home/ofuser/blueCFD/ofuser-of4/run/tutorials/incompressib
le/simpleFoam/airFoil2D

ofuser@machine_name MINGW64 OpenFOAM-4.x ~/blueCFD/ofuser-of4/run/tutorials/inco
mpressible/simpleFoam/airFoil2D
$
```

Interface on Windows

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Getting started with the interface (2/12)

Commands for file management (1/4):

`ls`	list directory contents
`ls -1`	same as above, but in a single column
`ls -al`	formatted listing with hidden files
`ll`	formatted listing, same as `ls -l`
`cd dirname`	go to directory `dirname`
`cd`	go to user home
`cd ..`	go back one directory
`pwd`	show current directory path
`mkdir dirname`	create directory `dirname`

Getting started with the interface (3/12)

Commands for file management (2/4):

`rm filename`	delete file `filename`
`rm -r dirname`	delete directory `dirname`
`rm -f filename`	force delete file `filename` (CAUTION)
`rm -rf dirname`	force delete directory `dirname` (CAUTION)

`cp filename1 filename2`

copy file `filename1` to file `filename2`

`cp -r dirname1 dirname2`

copy directory `dirname1` to directory `dirname2`

Getting started with the interface (4/12)

Commands for file management (3/4):

```
mv filename1 filename2
```

rename or move file `filename1` to file `filename2`

```
ln -s filename linkname
```

create symbolic link `linkname` to file `filename`

```
touch filename
```

create file `filename` or change times of file `filename`

```
less filename
```

interactively output the contents of file `filename`

Getting started with the interface (5/12)

Commands for file management (4/4):

`less filename`	output the contents of file `filename` "q" for ending the interactive mode
`head filename`	output the first 10 lines of file `filename`
`tail filename`	output the last 10 lines of file `filename`
`tail -f filename`	same as above, but updates continuously

Getting started with the interface (6/12)

Commands for system information:

`<command> --help`	
	if available, shows the available help for `<command>`
`date`	show the current date and time
`whoami`	display the user name you are logged in as
`uname -a`	show operating system kernel information
`df`	show disk usage
`du`	show directory space usage
`echo envab`	prints to screen value of system variable “envab”

optimises your technology

Watermatex 2019

the international water association

Getting started with the interface (7/12)

Commands for process management:

`ps`	display your currently active processes
`kill pid`	kill process with identification `pid` (CAUTION)
`jobs`	lists stopped or background jobs;
`bg`	resume a background job
`fg`	brings the most recent job to foreground
`fg n`	brings job `n` to the foreground

Getting started with the interface (8/12)

Commands for searching files and content (1/2):

`grep pattern filename`

search for `pattern` in `filename`

`grep -r pattern dirname`

search recursively for `pattern` in `dirname`

`command | grep pattern`

search for `pattern` in the output of `command`

`find dir -name pattern`

search for `pattern` in a directory hierarchy

Getting started with the interface (9/12)

Commands for searching files and content (2/2):

`find dir -name pattern | grep word`	search for `pattern` in a directory hierarchy and search within files that have been found for the `word` inside them
`which command`	locate a command `command`
`where command`	show possible locations the `command` name

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Getting started with the interface (10/12)

Text editors:

`vi file`	use text editor `vi` to edit file `file`
`nano file`	uses text editor `notepad2` to edit file `file`

(`nano` is available for MSys2 and is easier to use than `vi`, but it wasn't installed with blueCFD-Core)

Other text editors of note but not installed in blueCFD-Core:

• Notepad++	notepad-plus-plus.org
• Geany	geany.org
• Etc	your favorite

Getting started with the interface (11/12)

Command Line Navigation:

`Ctrl+a`	moves cursor to beginning of line
`Ctrl+e`	moves cursor to end of line
`Ctrl+→`	moves cursor to beginning of next word in the line
`Ctrl+←`	moves cursor to beginning of previous word in the line
`Ctrl+k`	deletes words until end of line from current cursor position
`Ctrl+u`	deletes words until the start of line from current cursor position
`Ctrl+y`	*pastes* the words that were deleted with `Ctrl+k/u`
`Alt+backspace`	deletes previous word in line from current cursor position
`Alt+F2`	Starts a new terminal window

FS DYNAMICS optimises your technology

Watermatex 2019 the international water association

Getting started with the interface (12/12)

Additional information about shells, commands and procedures on Linux can be obtained through:

- The Linux Documentation Project: www.tldp.org
- Linux Command website: linuxcommand.org

In the Linux Documentation Project website, we can also see a general introduction on Linux:

www.tldp.org/LDP/intro-linux/html/index.html

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

OpenFOAM®

Bird’s-eye view

optimises your technology

Watermatex 2019

the international water association

Section Contents

OpenFOAM Structure (1/3)

Important Environment Variables in OpenFOAM

$WM_PROJECT_DIR	path to the OpenFOAM installation
$WM_PROJECT_USER_DIR	user directory
$FOAM_TUTORIALS	tutorials
$FOAM_SRC	source code directory of OpenFOAM libraries
$FOAM_APP	source code directory of OpenFOAM applications
$FOAM_APPBIN	directory with the compiled OpenFOAM applications
$FOAM_USER_APPBIN	directory with the OpenFOAM applications created by the user
$FOAM_LIBBIN	directory with the compiled OpenFOAM libraries
$FOAM_USER_LIBBIN	directory with the OpenFOAM libraries created by the user
$FOAM_RUN	directory where the user can put his/her cases
echo *variable*	will show you the contents of environment variable, example: `echo $WM_PROJECT_DIR`

FS DYNAMICS optimises your technology

Watermatex 2019

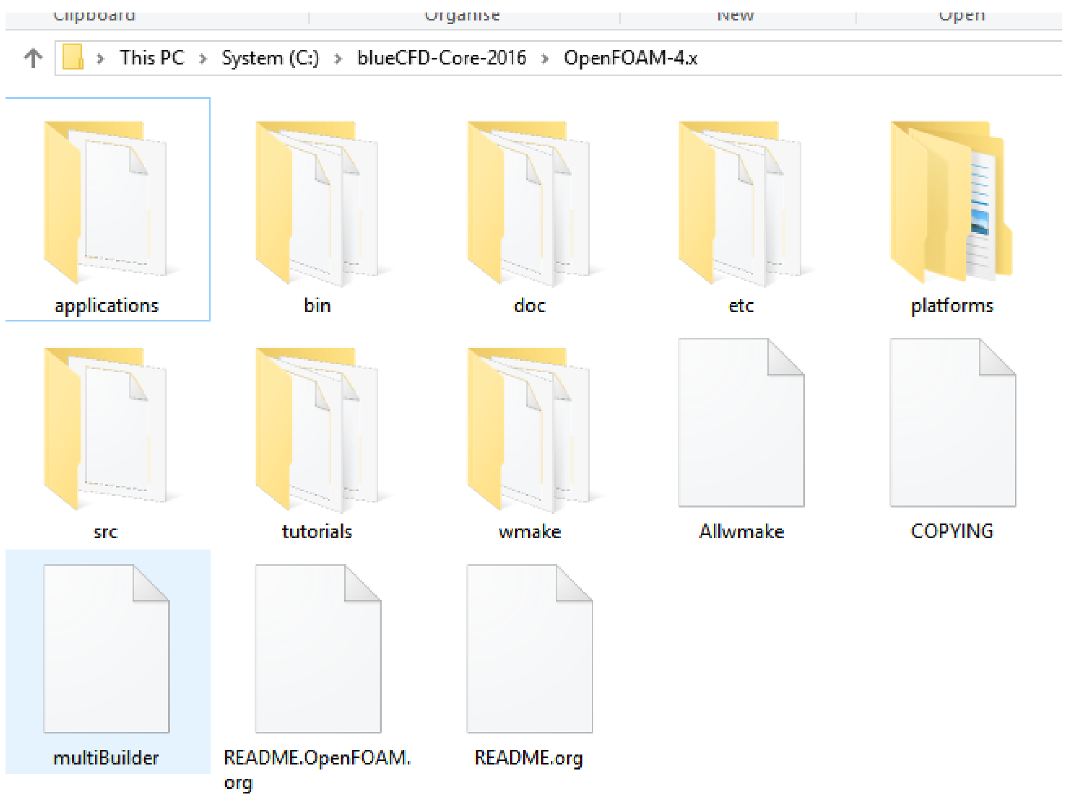
the international water association

OpenFOAM Structure (2/3)

Important Shell-Aliases in OpenFOAM

foam	cd $WM_PROJECT_DIR
app	cd $FOAM_APP
sol	cd $FOAM_SOLVERS
tut	cd $FOAM_TUTORIALS
util	cd $FOAM_UTILITIES
src	cd $FOAM_SRC/$WM_PROJECT
lib	cd $FOAM_LIBBIN
run	cd $FOAM_RUN
src	cd $FOAM_SRC
wmSet	. $WM_PROJECT_DIR/etc/bashrc
wmUnset	. $WM_PROJECT_DIR/etc/config/unset.sh

OpenFOAM Structure (3/3)

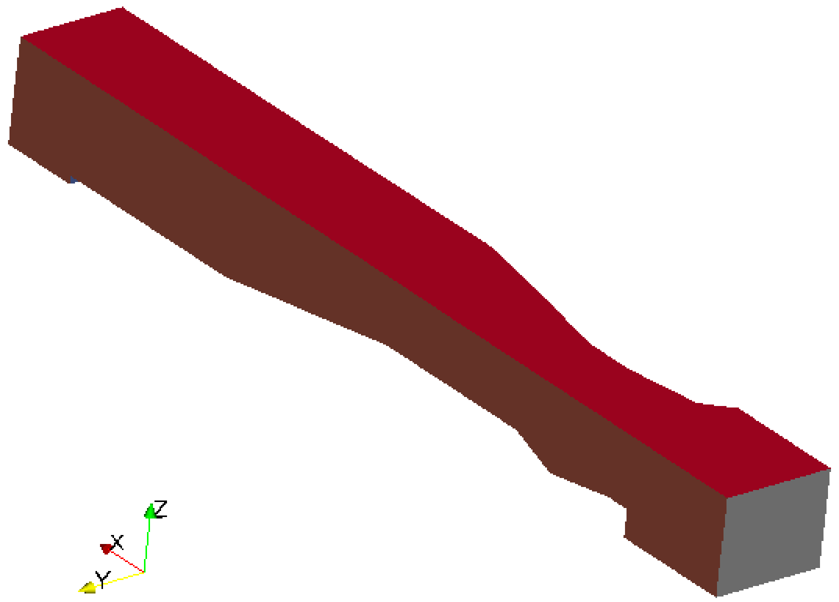


OpenFOAM's main folder structure:

- **applications** – source code for…
  - **solvers** – actual flow solvers
  - **test** – core function testing
  - **utilities** –utilities (i.e. everything else)
- **bin** – auxiliary scripts for using OpenFOAM
- **doc** –where the documentation is located
- **etc** – scripts for support files (shell environment, etc…)
- **platforms** – where the built binaries are placed
- **src** – the source code of the libraries
- **tutorials** – the tutorial cases
- **wmake** – script infrastructure for building OpenFOAM

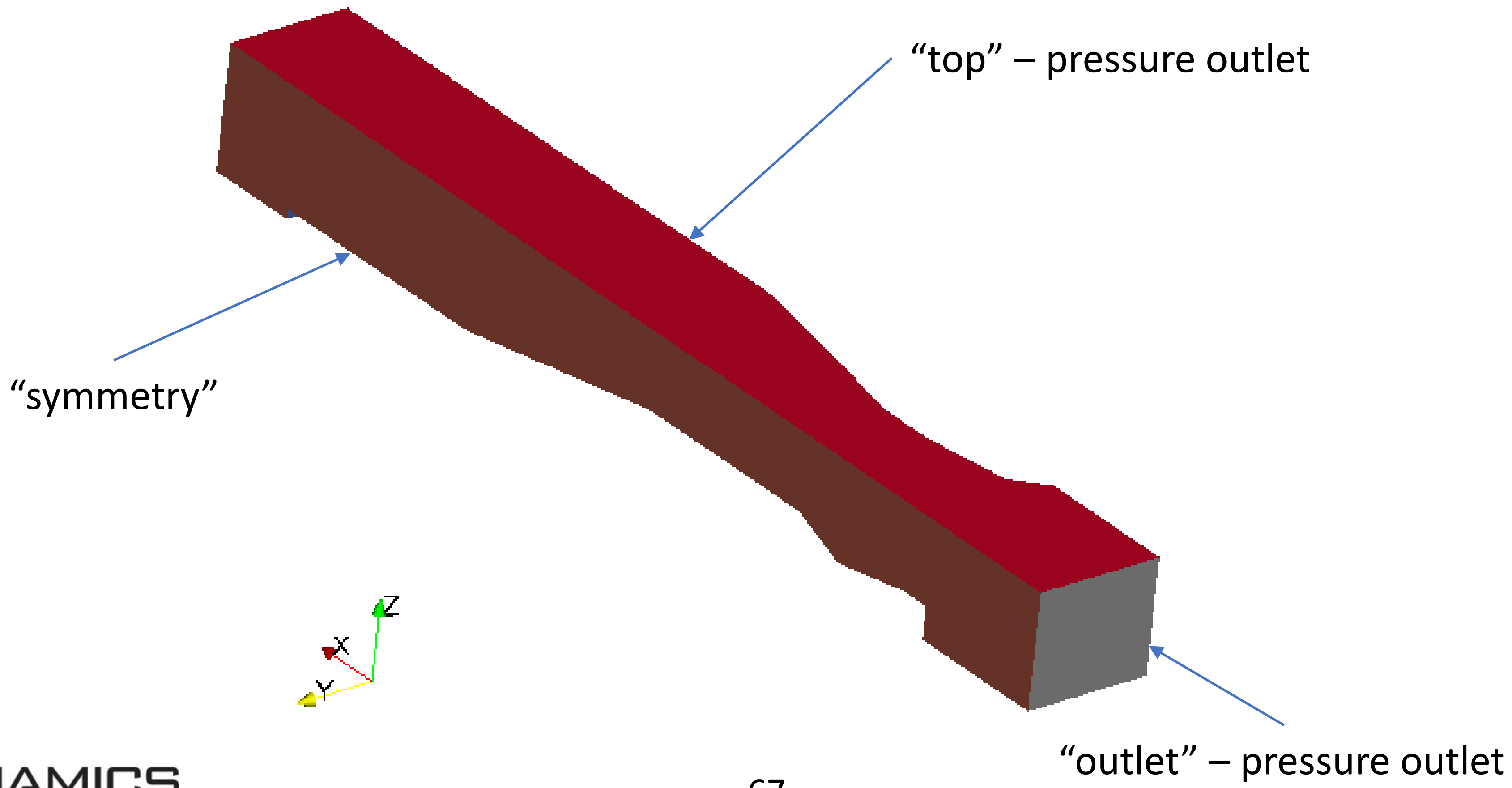
Example Case Overview (1/4)

- Case name: **halfParshall**
- Boundary conditions:
  - Inlet: **375 kg/s**
  - Bottom floor and side wall: **no-slip**
  - Outlet surfaces: **pressure outlet**
  - Symmetry plane surface: **symmetry**
- Fluid properties:
  - Water:
    - Density: **999 kg/m^3**
    - Dynamic Viscosity: **1.15E-3 Pa.s**
  - Air:
    - Density: **1.18 kg/m^3**
    - Dynamic Viscosity: **1.855E-5 Pa.s**



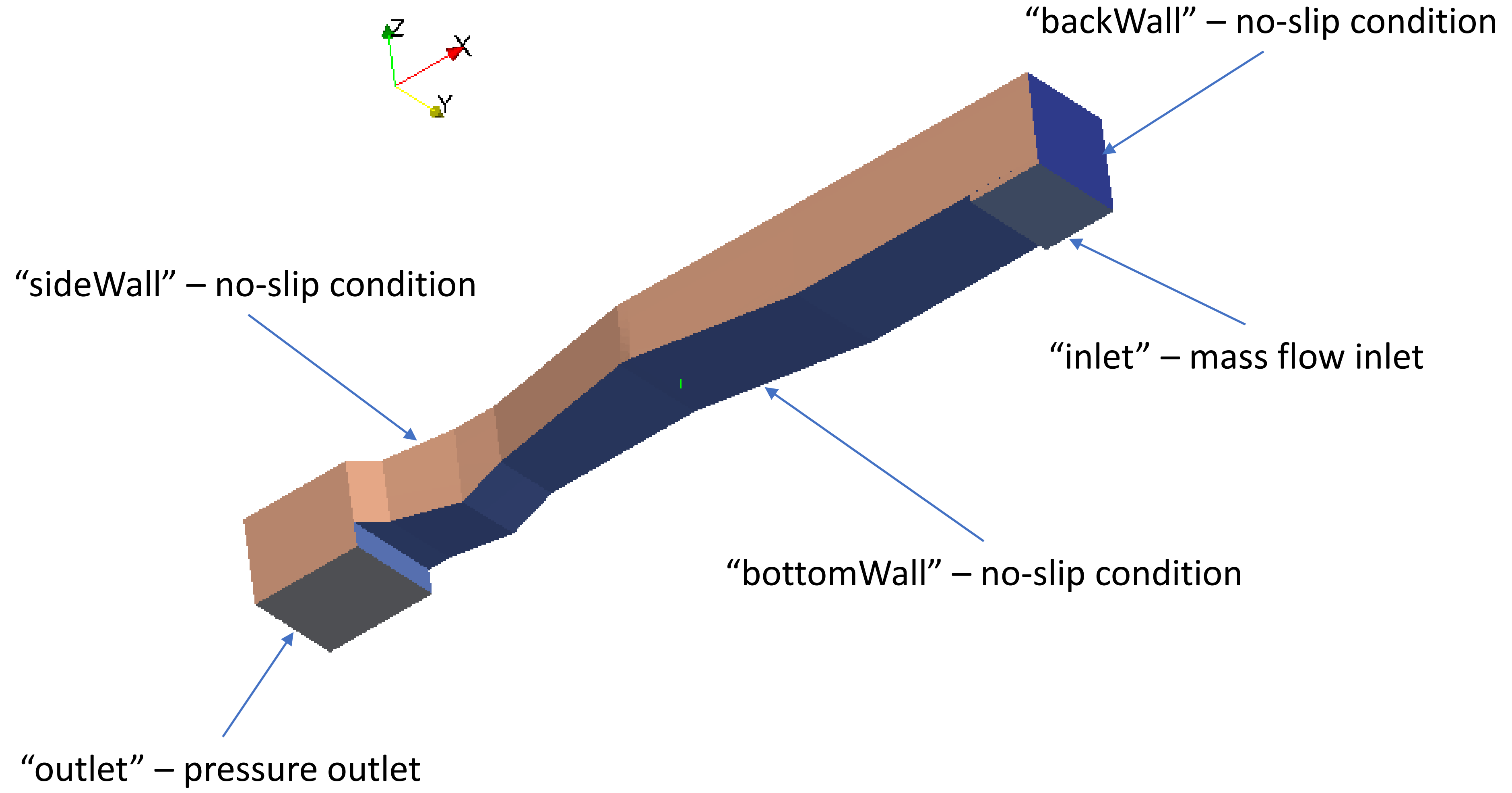
Example Case Overview (2/4)

- Solver type: VOF (volume of fluid)
- Time domain: transient
- Geometry (1/2):



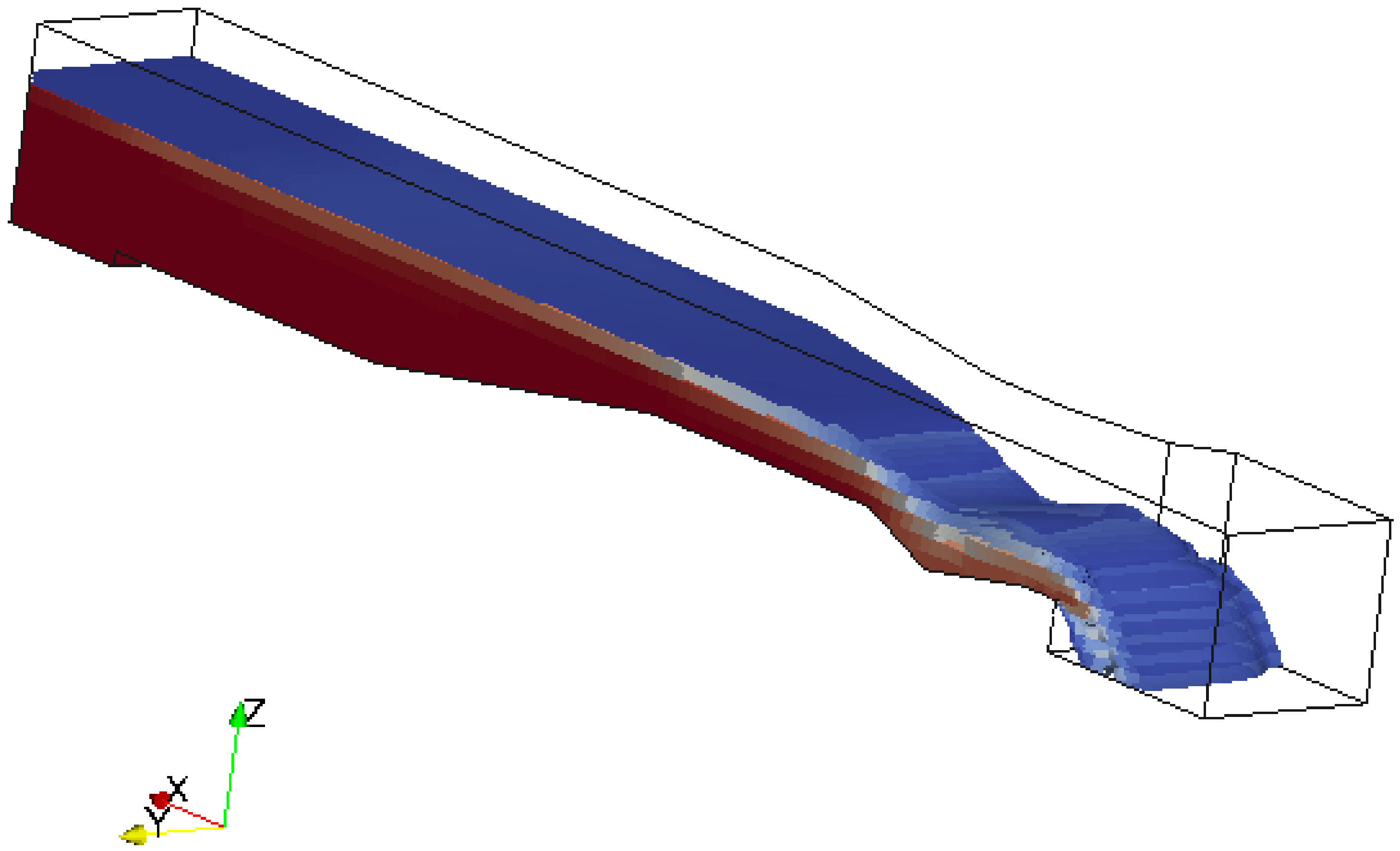
Example Case Overview (3/4)

- Geometry (2/2):


“backWall” – no-slip condition
“sideWall” – no-slip condition
“inlet” – mass flow inlet
“bottomWall” – no-slip condition
“outlet” – pressure outlet
Z
X
Y

Example Case Overview (4/4)

- Objective:



Simulation Case Structure (1/6)

The case definition is clear. Now what?

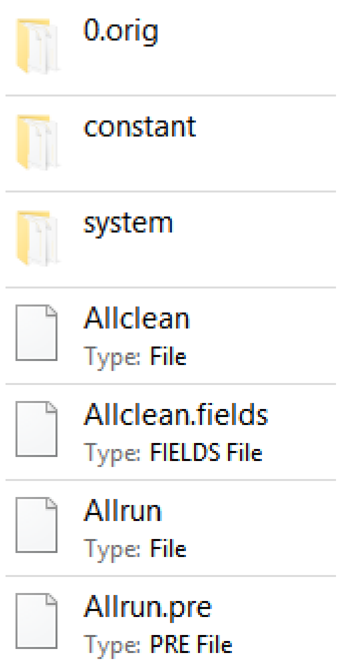
- We select the solver which suits the case characterization.
  - On this case: `interFoam`
- Search a tutorial case that uses that same solver, either from the default tutorial collection (`$FOAM_TUTORIALS`) or through a web search, and copy it across.
- Start adjusting settings (boundary conditions, initialization and numerical parameters) to suit our needs.

Naturally, for meshing to be done, the **CAD** has to be available as well.

Simulation Case Structure (2/6)

Folder structure in OpenFOAM for our case:

- **halfParshall**
  - **0.orig** – BC's and initialization
    - **U** – velocity field
    - **p** – pressure field
    - *etc…*
  - **constant** – e.g. physical properties
    - **polymesh** – polyhedral mesh files
    - **triSurface** – geometrical models
  - **system** – numerics and run-time control
  - *time directories* – examples: 0, 0.1, 1, 2, 3 and so on.



0.orig

constant

system

Allclean
Type: File

Allclean.fields
Type: FIELDS File

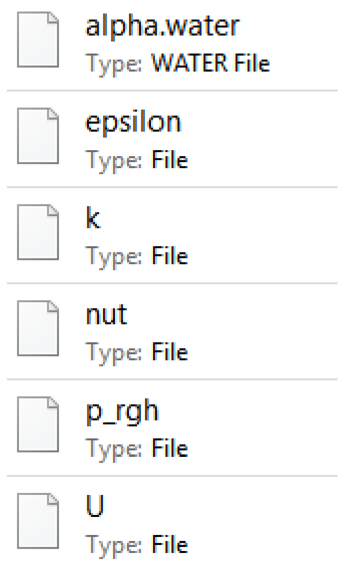
Allrun
Type: File

Allrun.pre
Type: PRE File

Simulation Case Structure (3/6)

halfParshall/0.orig:

- **U** – velocity field
- **p_rgh** – pressure field
- **alpha.water** – phase fraction field
  - 1 = 100% water
  - 0 = 100% not water (air in our case)
- **epsilon** – turbulent dissipation rate field
- **k** – turbulent kinetic energy field
- **nut** – turbulent dynamic viscosity field



```
alpha.water
Type: WATER File
epsilon
Type: File
k
Type: File
nut
Type: File
p_rgh
Type: File
U
Type: File
```

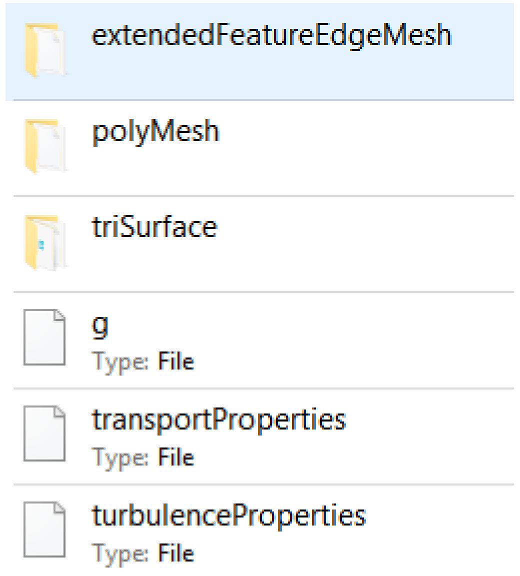
Requirement before running the solver:

```
cp -r 0.orig 0
```

Simulation Case Structure (4/6)

halfParshall/constant:

- **g** – gravity configuration (acceleration vector)
- **transportProperties** – physical properties of the fluids
- **turbulenceProperties** – turbulence model type: RAS or LES and respective configuration
- **polyMesh**
    - all other files are respective to the mesh, i.e. automatically created, including:
        - **boundary** – geometrical boundary conditions
- **triSurface**
    - **halfParshall.org.stl** – original geometrical model, in STL format


extendedFeatureEdgeMesh
polyMesh
triSurface
g
Type: File
transportProperties
Type: File
turbulenceProperties
Type: File

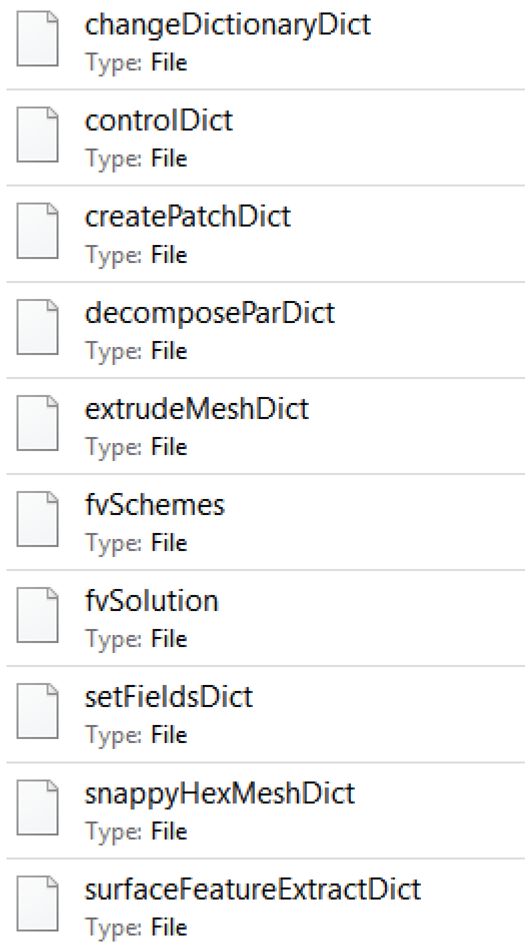
Simulation Case Structure (5/6)

halfParshall/system – essential configuration files:

- **controlDict** – runtime controls (start/stop time, etc...)
- **fvSchemes** –discretization schemes
- **fvSolution** – linear equation solvers and algorithms

Application-specific files:

- **blockMeshDict** – dictionary file for **blockMesh**
- **changeDictionaryDict** – to manipulate dictionary files
- **createPatchDict** – to create/remove/manipulate patches
- **decomposeParDict** – subdomain decomposition
- **extrudeMeshDict** – for extruding the mesh
- **setFieldsDict** – for manipulating the fields
- **snappyHexMeshDict** – dictionary for **snappyHexMesh**
- **surfaceFeatureExtractDict** – for calculating feature edges



changeDictionaryDict
Type: File

controlDict
Type: File

createPatchDict
Type: File

decomposeParDict
Type: File

extrudeMeshDict
Type: File

fvSchemes
Type: File

fvSolution
Type: File

setFieldsDict
Type: File

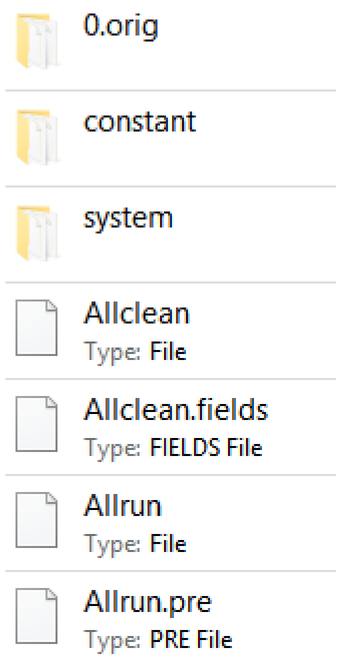
snappyHexMeshDict
Type: File

surfaceFeatureExtractDict
Type: File

Simulation Case Structure (6/6)

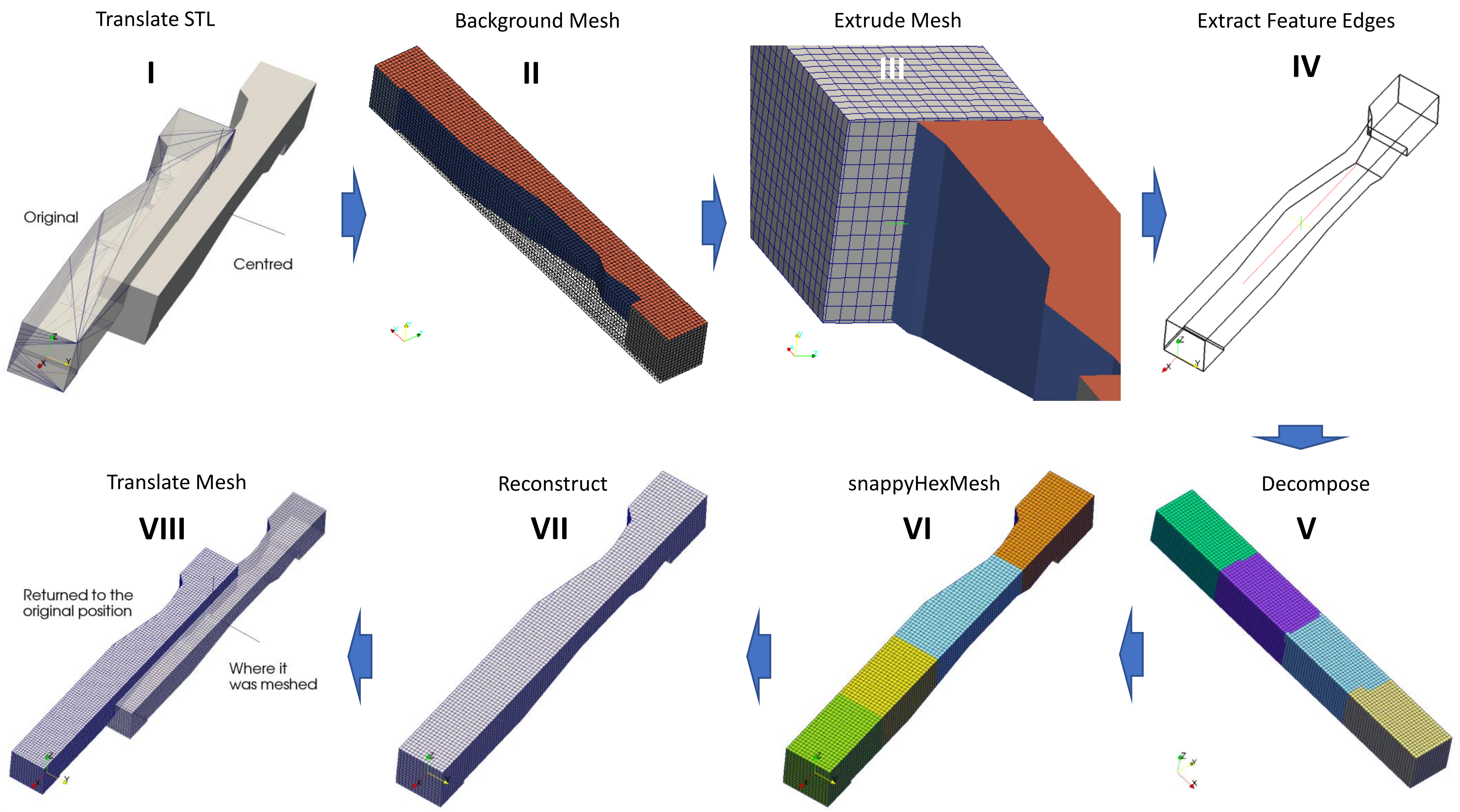
halfParshall – files in the case's root folder:

- Scripts for setting up and running the case:
  - **Allrun** – will run all steps, also calls **Allrun.pre**
  - **Allrun.pre** – will preprocess and generate the mesh
- Scripts for resetting the case to the original state:
  - **Allclean** – will reset (clean up) the whole case
  - **Allclean.fields** – will only remove the time snapshots



These scripts are manually created, nonetheless several examples for these files are available in OpenFOAM's "`tutorials`" folder.

Mesh Generation (1/19)



Mesh Generation (2/19)

Meshing steps overview – Contents (*nano it*) of the **Allrun.pre** script (1/2):

```
runApplication surfaceTransformPoints -translate '(-4.25 0.687 -0.55)' \
    constant/triSurface/halfParshall.org.stl \
    constant/triSurface/halfParshall.stl
```

I

```
runApplication blockMesh
runApplication extrudeMesh
runApplication surfaceFeatureExtract
```

II, III, IV

```
echo "decompositionMethod  scotch;" > system/decomposeParDict.method
runApplication -s 1  decomposePar
echo "decompositionMethod  ptscotch;" > system/decomposeParDict.method
```

V

Mesh Generation (3/19)

Meshing steps overview – Contents of the **Allrun.pre** script (2/2):

```
runParallel snappyHexMesh -overwrite
```
VI

```
runApplication reconstructParMesh -constant

runApplication createPatch -overwrite
```
VII

```
runApplication transformPoints -translate '(4.25 -0.687 0.55)'

runApplication checkMesh -constant

runApplication changeDictionary -enableFunctionEntries
```
VIII

optimises your technology

Watermatex 2019

the international water association

Mesh Generation (4/19)

Meaning of each step (1/5):

- **runApplication** and **runParallel** – these are auxiliary script functions, for logging the execution of the application.
- **surfaceTransformPoints** – used for centring the geometry onto the world referential.
- **blockMesh** – generates the base mesh, which wraps our geometry within it, acting as a bounding box. Requires the file "`system/blockMeshDict`".
- **extrudeMesh** – in our case, we use it to add one additional cell layer around the original base mesh, for improving the wrapping around our geometry. Requires the file "`system/extrudeMeshDict`".

Mesh Generation (5/19)

Meaning of each step (2/5):

- **surfaceFeatureExtract** – will calculate the feature edges on our STL file. Requires these files:
  - "`system/surfaceFeatureExtractDict`"
  - "`constant/triSurface/halfParshall.stl`"
- **decomposePar** – will decompose our existing mesh so far into 4 subdomains, so that we can mesh with 4 processes in parallel. Requires the file "`system/decomposeParDict`".
  - The option "`-s 1`" is for appending the suffix "`.1`" to the log file name, because `decomposePar` will be used a second time later on.

Mesh Generation (6/19)

Meaning of each step (3/5):

- **snappyHexMesh** – this is the main mesh generator we will use, which takes the base mesh we created and it will:
  - refine the mesh accordingly to our settings;
  - remove the cells that don’t matter from the mesh, in this case, the cells that are outside of our geometry;
  - morph and cut (i.e. *snap*) the mesh’s surface onto the surfaces of our geometry.

All of the above settings are defined in the file “`system/snappyHexMeshDict`”.

optimises your technology

Watermatex 2019

the international water association

Mesh Generation (7/19)

Meaning of each step (4/5):

- **reconstructParMesh** – this will reconstruct the resulting 4 subdomain meshes into a single mesh.
- **createPatch** – it will clean up the list of patches in our mesh, because the original patches from the base mesh would otherwise remain present, with 0 faces assigned.
- **transformPoints** – used in our case for moving the whole mesh back into the original position of the original geometry.
- **checkMesh** – used for keeping a record of the characteristics of the mesh, including any diagnosed flaws.

Mesh Generation (8/19)

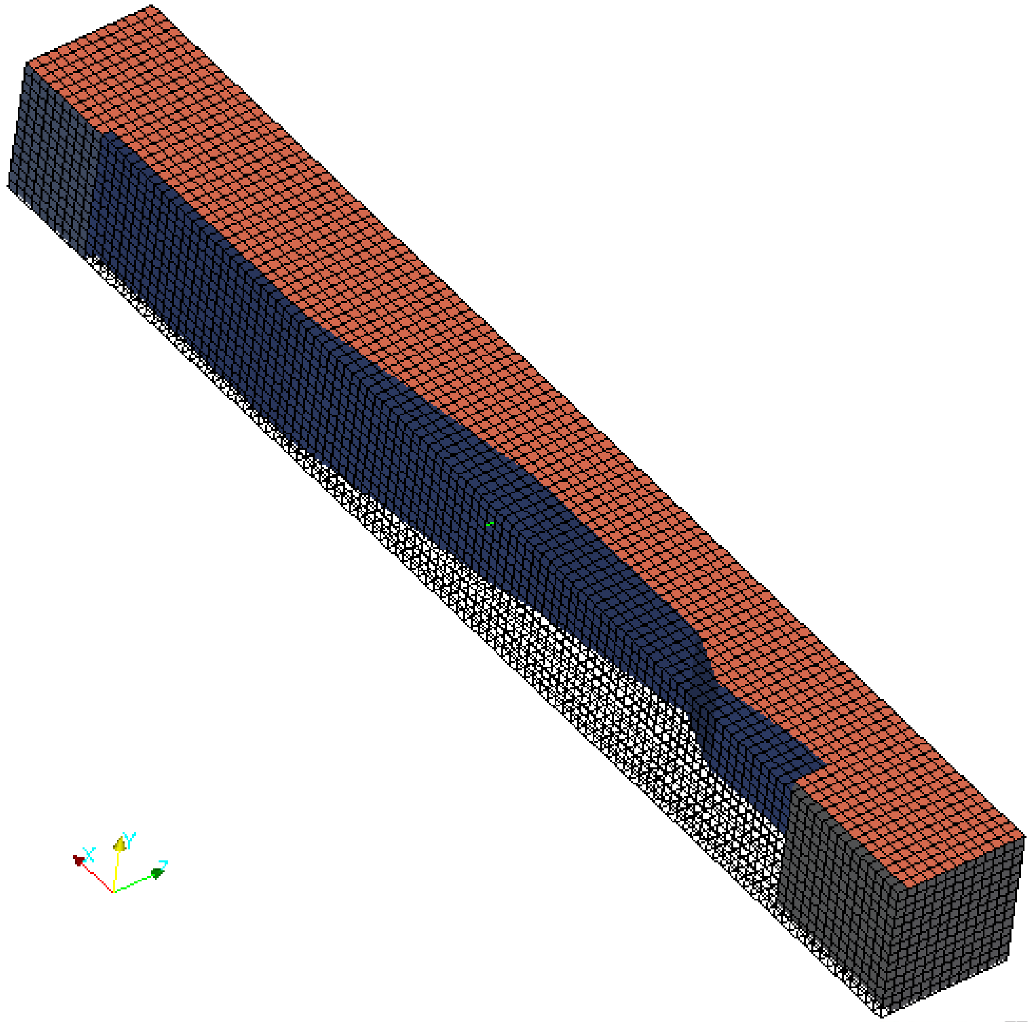
Meaning of each step (5/5):

- **changeDictionary** – in our case we use it for changing the type of surface boundary we want for each patch, namely if each surface is a “patch”, “wall” or “symmetry”.
- The 2 commands that use `echo` are for defining the decomposition method to be used. This is because:
  - the “`scotch`” method is designed to work well in serial mode;
  - the “`ptscotch`” method is designed to work well in parallel mode.
- The file “`system/decomposeParDict.method`” is used by the main dictionary file “`system/decomposeParDict`”.

Mesh Generation (9/19)

Seeing the mesh (1/5) - mesh done with **blockMesh**:

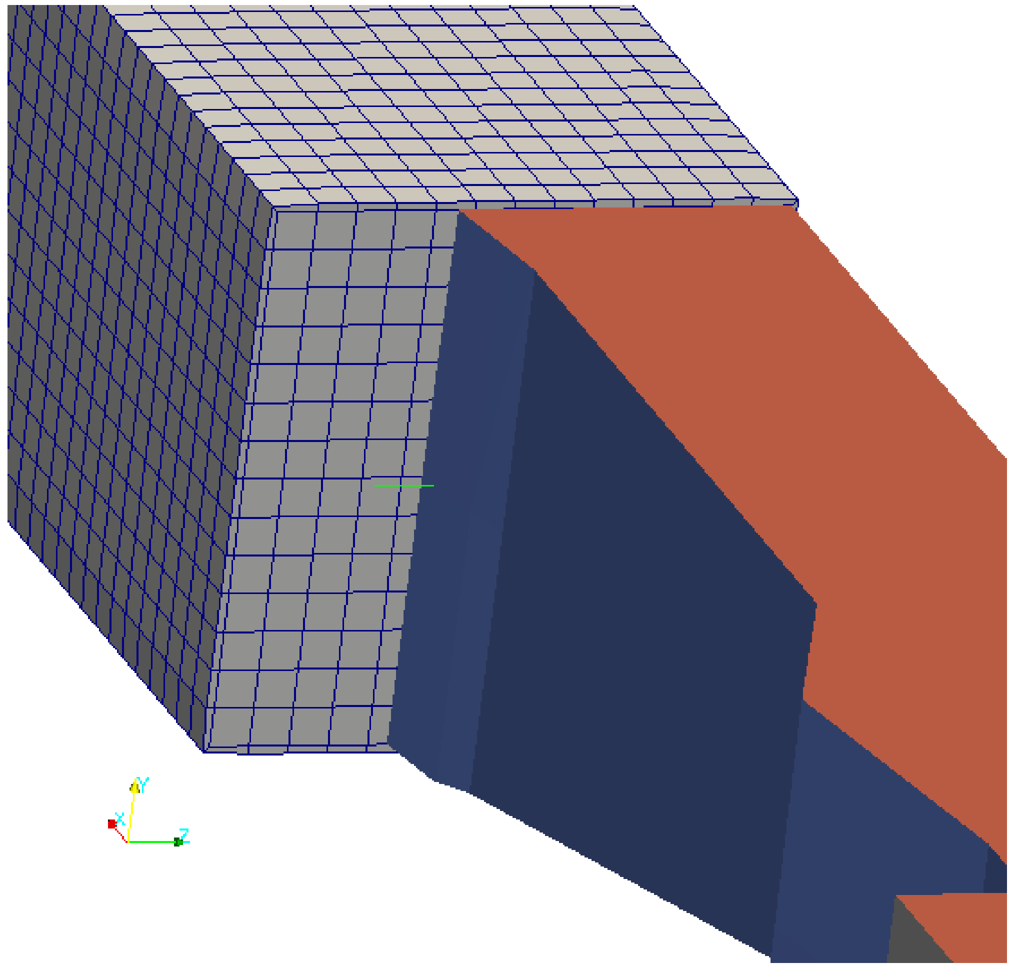
As shown on the right, it is a somewhat tight bounding box around our geometry, which is also why we refer to this as the "background mesh".



Mesh Generation (10/19)

Seeing the mesh (2/5) - the extruded mesh:

This is a detail view of the inside of the mesh after the extrusion is done. This will make it easier for **snappyHexMesh** to *see* the geometry.

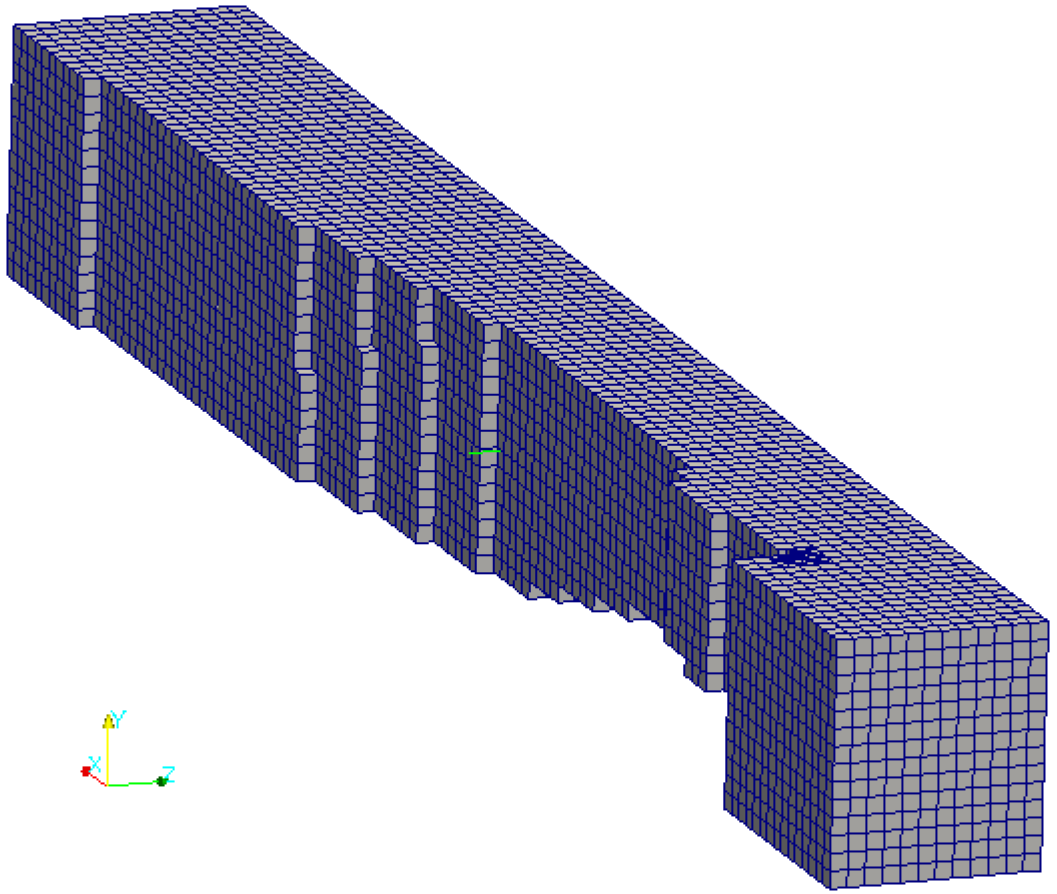


Mesh Generation (11/19)

Seeing the mesh (3/5) – 1st step of **snappyHexMesh**:

This is the result of the *castellation* step, where it:

1. Refines the mesh where asked to.
2. Removes the cells that are irrelevant for our final mesh.

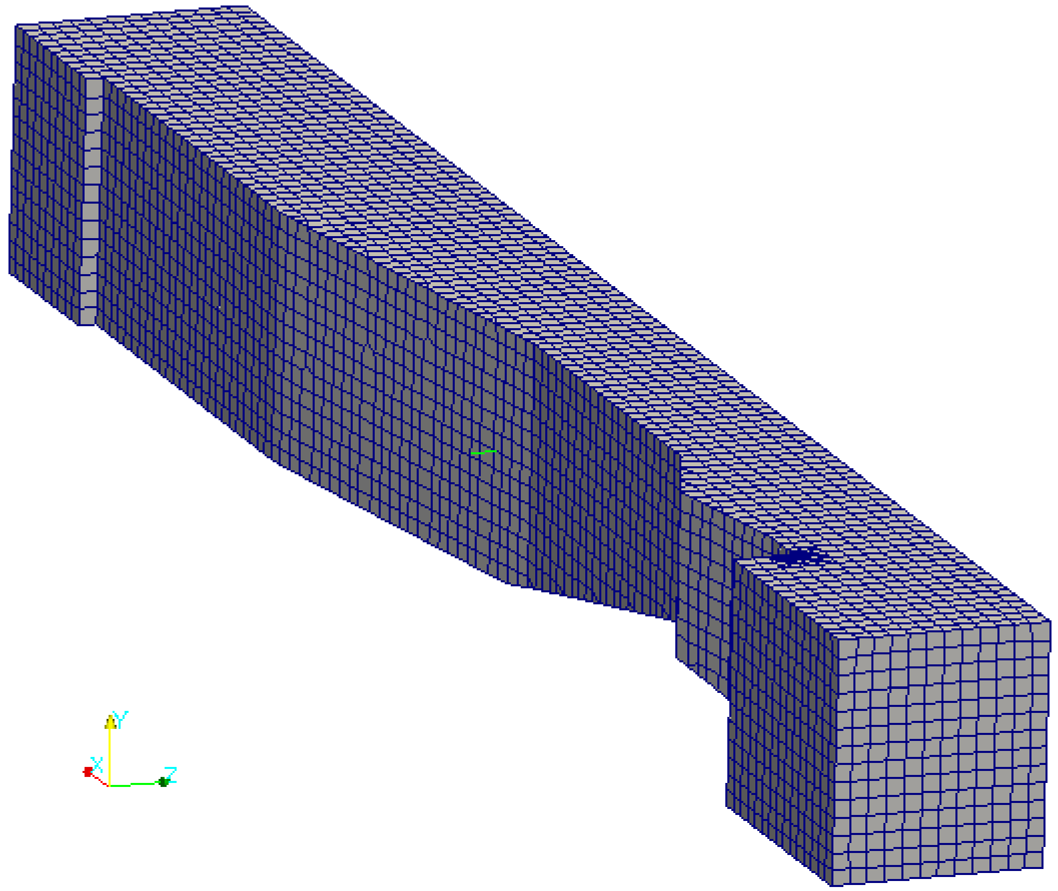


Mesh Generation (12/19)

Seeing the mesh (4/5) – 2nd step:

This is the result of the *snap* step, where it:

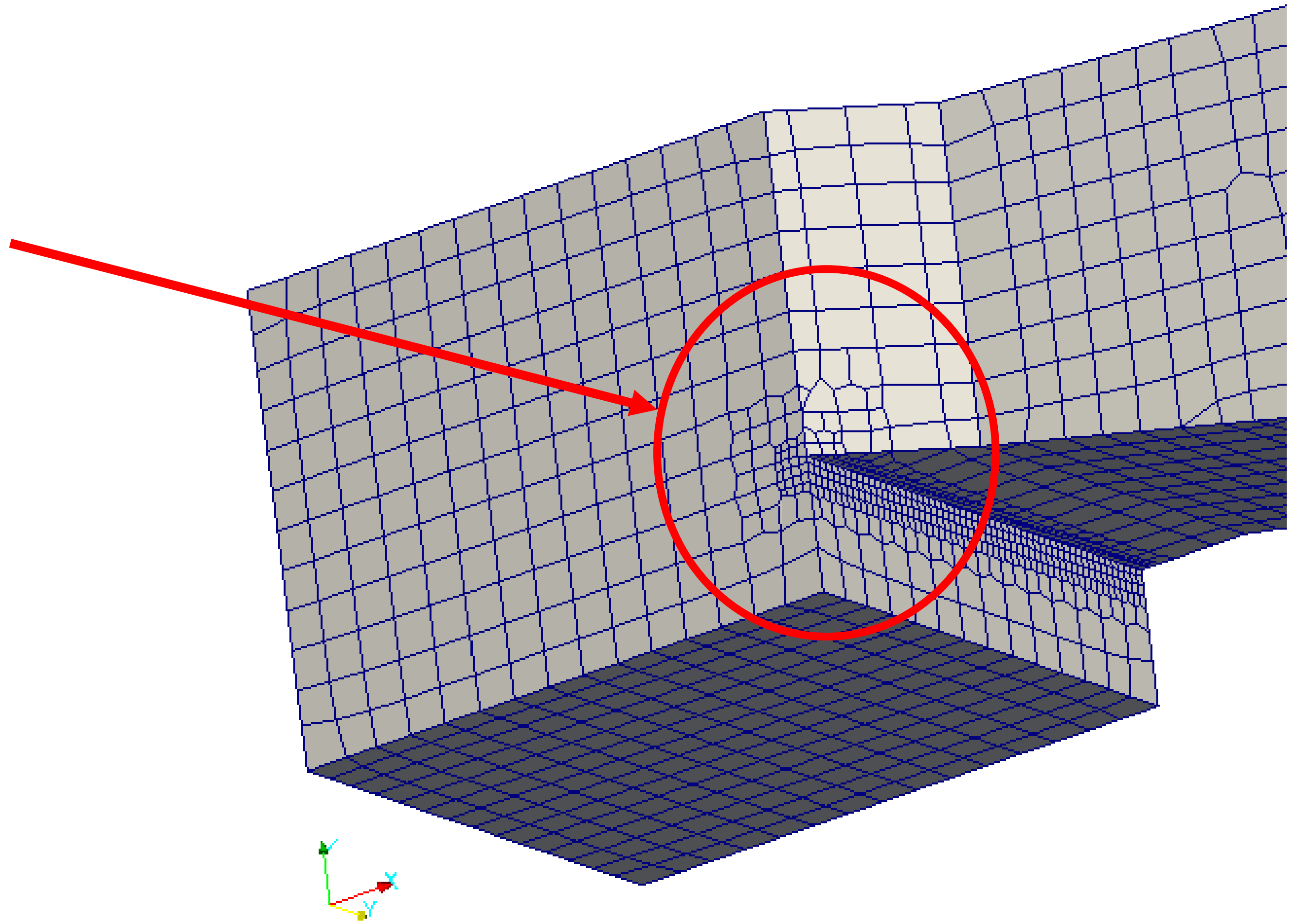
1. Cuts cells that overlap the geometry.
2. Morphs (snaps) the mesh onto the surface.



Mesh Generation (13/19)

Seeing the mesh (5/5) – refinement detail:

This is why we needed more refinement near the outlet.



Mesh Generation (14/19)

Note: This header is common to all of OpenFOAM's dictionary files:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant/polyMesh";
    object      blockMeshDict;
}
```

This is part of OpenFOAM's open file format standard, so that special data readers aren't needed for human manipulation.

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Mesh Generation (15/19)

Highlights of **blockMeshDict**:

```
convertToMeters 1;

vertices
(
   (-5.75 -0.686883 -0.65)
   (5.75  -0.686883 -0.65)
   (5.75   0.687 -0.65)
   (-5.75  0.687 -0.65)
   (-5.75 -0.686883 0.65)
   (5.75  -0.686883 0.65)
   (5.75   0.687 0.65)
   (-5.75  0.687 0.65)
);

blocks
(
   hex (0 1 2 3 4 5 6 7) (115 14 13) simpleGrading (1 1 1)
);
```

```
patches
(
   patch maxX
   ( (1 2 6 5) )
   patch minX
   ( (0 4 7 3) )
   patch maxY
   ( (3 7 6 2) )
   patch minY
   ( (1 5 4 0) )
   patch maxZ
   ( (4 5 6 7) )
   patch minZ
   ( (0 3 2 1) )
);
```

Mesh Generation (16/19)

Highlights of **snappyHexMeshDict** (1/4):

```
// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       false;

geometry
{
  "halfParshall.stl"
  {
    type triSurfaceMesh;
    regions
    {
      backWall
      {
        name backWall;
      }
       […]
```

```
 […]

  top
  {
    name top;
  }
 }
}

refinementBox
{
 type    searchableBox;
 min     (-4.35 -1 -0.30);
 max     (-4.20  1 -0.15);
}

};
```

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Mesh Generation (17/19)

Highlights of **snappyHexMeshDict** (2/4):

```
// Settings for the castellatedMesh generation.
castellatedMeshControls
{
  […]
    features
    ( {
        file "halfParshall.eMesh";
        level 0;
      } );

    refinementSurfaces
    {
      "halfParshall.stl"
      {
        level (0 0);
  […]
```

```
[…]
    regions
    {
      backWall
      {
        level (0 0);
      }
[…]
      top
      {
        level (0 0);
      }
    }
  }
}

[…]
```

Mesh Generation (18/19)

Highlights of **snappyHexMeshDict** (3/4):

```
 […]

  refinementRegions
  {
    refinementBox
    {
      mode inside;
      levels ((1e-15 2));
    }
  }

  locationInMesh (-1.212134e+000 8.290353e-003 -4.588275e-002);
}

//end of castellatedMeshControls
```

optimises your technology

Watermatex 2019

the international water association

Mesh Generation (19/19)

Highlights of **snappyHexMeshDict** (4/4):

```
// Settings for the snapping.
snapControls
{
  nSmoothPatch 3;

  tolerance 1.0;

  nSolveIter 30;

  nRelaxIter 5;
}
```

optimises your technology

Watermatex 2019

the international water association

Preprocessing (1/9)

Model Properties (1/2) – In file `constant/transportProperties`:

```
transportModel  Newtonian;
phases (water air);
water
{
    transportModel  Newtonian;
    nu              nu    [ 0 2 -1 0 0 0 0 ] 1.131131e-006;
    rho             rho   [ 1 -3 0 0 0 0 0 ] 9.990000e+002;
    mu              mu    [ 1 -1 -1 0 0 0 0 ] 1.130000e-003;
}
air
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ]  1.572e-05;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1.18;
    mu              mu    [ 1 -1 -1 0 0 0 0 ] 1.855e-05;
}
sigma           sigma [ 1 0 -2 0 0 0 0 ] 0;
```

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Preprocessing (2/9)

Model Properties (2/2) – Where:

- *nu* – kinematic viscosity (m^2/s)
- *mu* – dynamic viscosity (kg.m/s)
- *rho* - volumetric mass density (kg/m^3)
- *sigma* - surface tension (kg/s^2 or N/m)
- *phases* – the list of named phases present in the domain.
  - The names do not strictly define the fluid their representing, they are only for identification purposes.

optimises your technology

Watermatex 2019

the international water association

Preprocessing (3/9)

Boundary Conditions (1/2) – In a nutshell:

- 6 field files: *U*, *alpha.water*, *epsilon*, *k*, *nut*, *p_rgh*
- 4 major groups of boundary conditions per field:
  - Inlet – assigned to the “inlet” surface
  - Outlet – assigned to the “outlet” and “top” surfaces
  - Wall – assigned to the “backWall”, “bottomWall”, “sideWall” surfaces
  - Symmetry – assigned to the “symmetry” surface

Preprocessing (4/9)

Boundary Conditions (2/2) – For example, *U* field file:

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0.0 0.0 0.0);
boundaryField
{
    backWall
    {
        type                fixedValue;
        value               uniform (0.0 0.0 0.0);
    }
…
    inlet
    {
      type                    flowRateInletVelocity;
      massFlowRate            375;
      rho                     rho;
      rhoInlet                999.0;
    }
```

Preprocessing (5/9)

In the file `system/fvSolution` (1/3):

This dictionary file was designed to handle the settings for the linear equation solvers and the algorithms to be used by a solver application, e.g. **interFoam**.

Starting with the linear equation solvers, these are configured inside block list:

```
solvers
{
   …
}
```

The next few slides show one example.

Preprocessing (6/9)

In the file `system/fvSolution` (2/3):

For configuring the linear equation solvers for the fields that start with “alpha.water”, the example case we are using has the following settings:

```
"alpha.water.*"
{
  nAlphaCorr       2;
  nAlphaSubCycles 1;
  cAlpha           1;

  MULESCorr        yes;
  nLimiterIter     3;

  solver           smoothSolver;
  smoother         symGaussSeidel;
  tolerance        1e-8;
  relTol           0;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Preprocessing (7/9)

In the file `system/fvSolution` (3/3):

Regarding the algorithm, **interFoam** uses PIMPLE, defined at the same levels as “solvers”:

```
solvers
{
  …
}

PIMPLE
{
  momentumPredictor   no;
  nOuterCorrectors    1;
  nCorrectors         3;
  nNonOrthogonalCorrectors 1;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019 the international water association

Preprocessing (8/9)

Finite volume discretization schemes in `system/fvSchemes` (1/2):

```
ddtSchemes
{
  default  Euler;
}

gradSchemes
{
  default  Gauss linear;
}

[…] (divSchemes in next slide)

laplacianSchemes
{
  default  Gauss linear corrected;
}
```

```
interpolationSchemes
{
  default   linear;
}

snGradSchemes
{
  default   corrected;
}

fluxRequired
{
  default   no;
  p_rgh;
  pcorr;
  alpha.water;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Preprocessing (9/9)

Finite volume discretization schemes in `system/fvSchemes` (1/2):

These usually come after `gradSchemes`:

```
divSchemes
{
  default                          none;
  div(rhoPhi,U)      Gauss upwind;
  div(phi,alpha)     Gauss upwind;
  div(phirb,alpha)   Gauss upwind;
  div(phi,k)         Gauss upwind;
  div(phi,epsilon)   Gauss upwind;
  div((muEff*dev(T(grad(U))))) Gauss linear;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Simulation (1/4)

Simulation steps overview – as simple as looking into the contents of the `Allrun` script:

```
./Allrun.pre

cp -r 0.orig 0

echo "decompositionMethod  scotch;" > system/decomposeParDict.method
runApplication -s 2  decomposePar -force

runParallel renumberMesh 4 -overwrite
#runParallel setFields 4
runParallel interFoam 4

runApplication reconstructPar
```

In the next slides we will see what each does…

Simulation (2/4)

Simulation steps (1/3):

- `Allrun.pre` – We learned about it in the meshing section.
- `cp -r 0.orig 0` – Deploy initial fields for the time step "0".
- `decomposePar -force` – This will re-decompose our mesh, along with the fields. We could have used the option "`-fields`", but we want to make sure that the mesh is well balanced, which might not be the case when `snappyHexMesh` is finished.
- `renumberMesh` – This application is meant to optimize how the cells (and respective data) in the mesh are organized, so that the equation matrices have a diagonal bandwidth as small as possible. Performance improvements can reach 30% less runtime.

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Simulation (3/4)

Simulation steps (2/3):

- `Allrun.pre` – We learned about it in the meshing section.
- `setFields` – Not used in our example case, but this is one of the reasons as to why we need the folder "`0.orig`" to be created separately, since running it will change the field files in the "`0`" folder.
- `interFoam` – This is the solver used in this case. Quoting the source:

  Solver for 2 incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach.

  The momentum and other fluid properties are of the "mixture" and a single momentum equation is solved.

  Turbulence modelling is generic, i.e. laminar, RAS or LES may be selected.

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Simulation (4/4)

Simulation steps (3/3):

- `reconstructPar` – For reconstructing the time steps that were generated while running in parallel.

Beyond this, comes the need for monitoring the output of the log file… which will seem fairly cryptic at first glance. For example:

```
GAMG:  Solving for p_rgh, Initial residual = 1, Final residual = 0.020995727, No Iterations 4
```

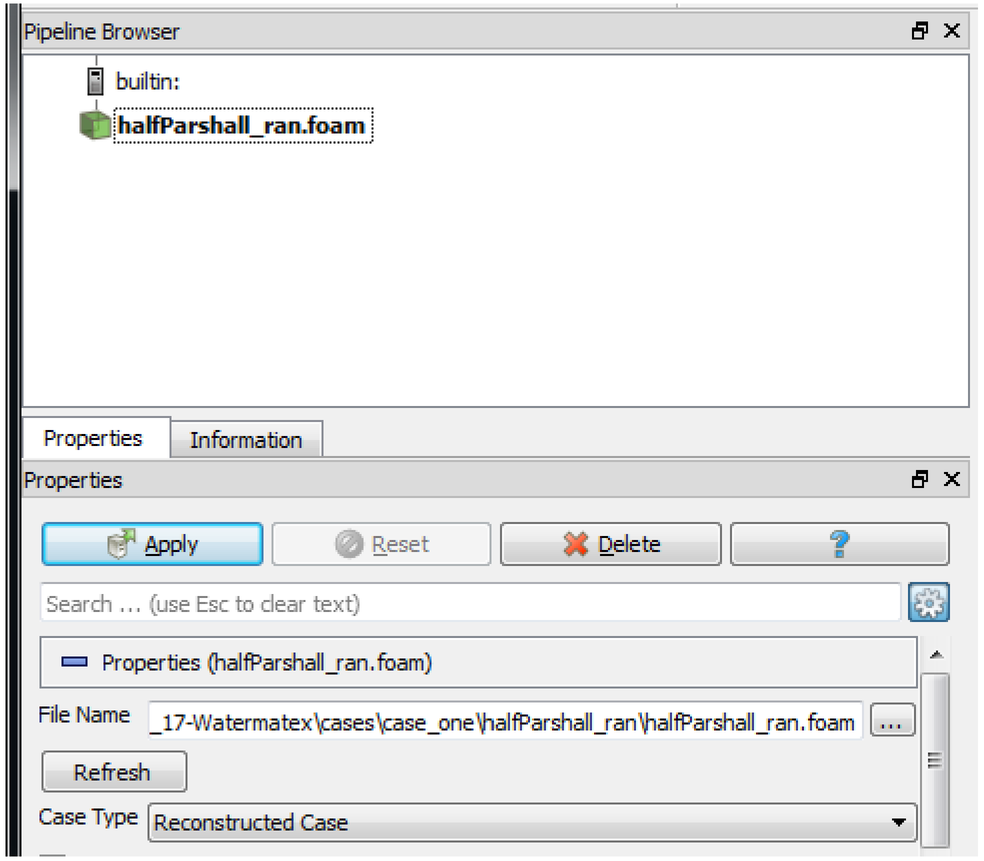
Not to worry, there is more than one way to plot the values that matter to us.

Post-processing (1/3)

The easiest post-processing is done by using ParaView, which can be launched with the script:

```
paraFoam
```

ParaView will start and show on the left side of the window, something similar to the one on the right. Clicking on the "Apply" button will load the case.



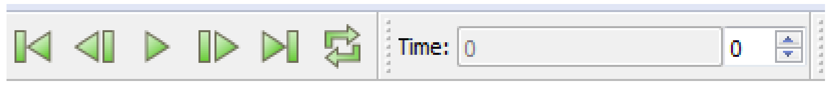
File Name: _17-Watermatex\cases\case_one\halfParshall_ran\halfParshall_ran.foam

Case Type: Reconstructed Case

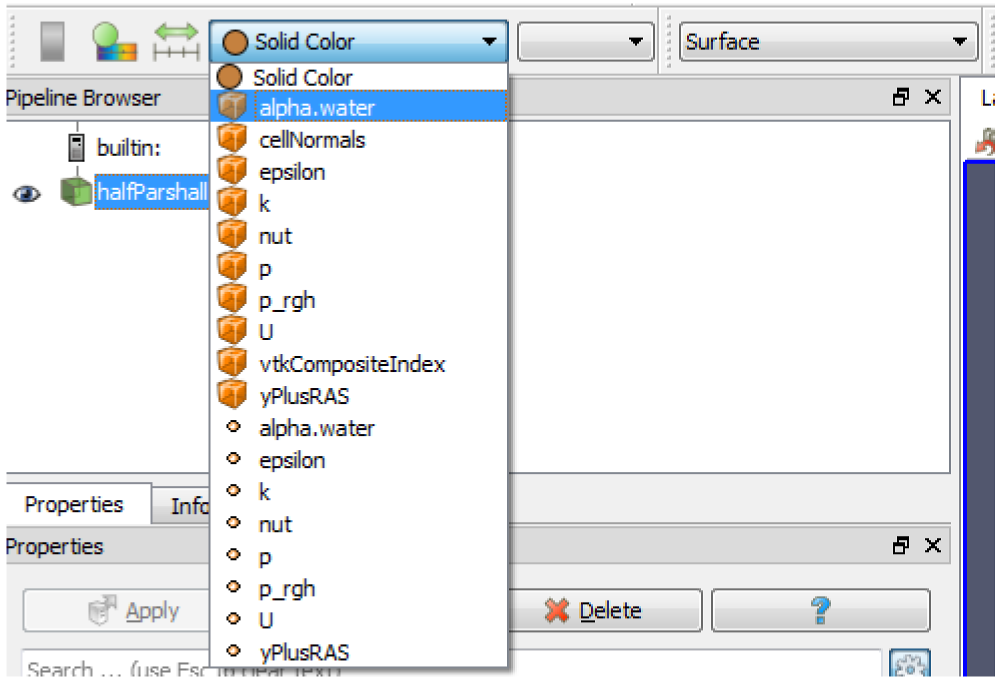
Post-processing (2/3)

The mouse controls with the 3D view are similar to most 3D CAD software, although the actions done by each button may be switched.

These are the time controls:

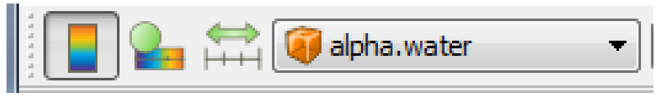


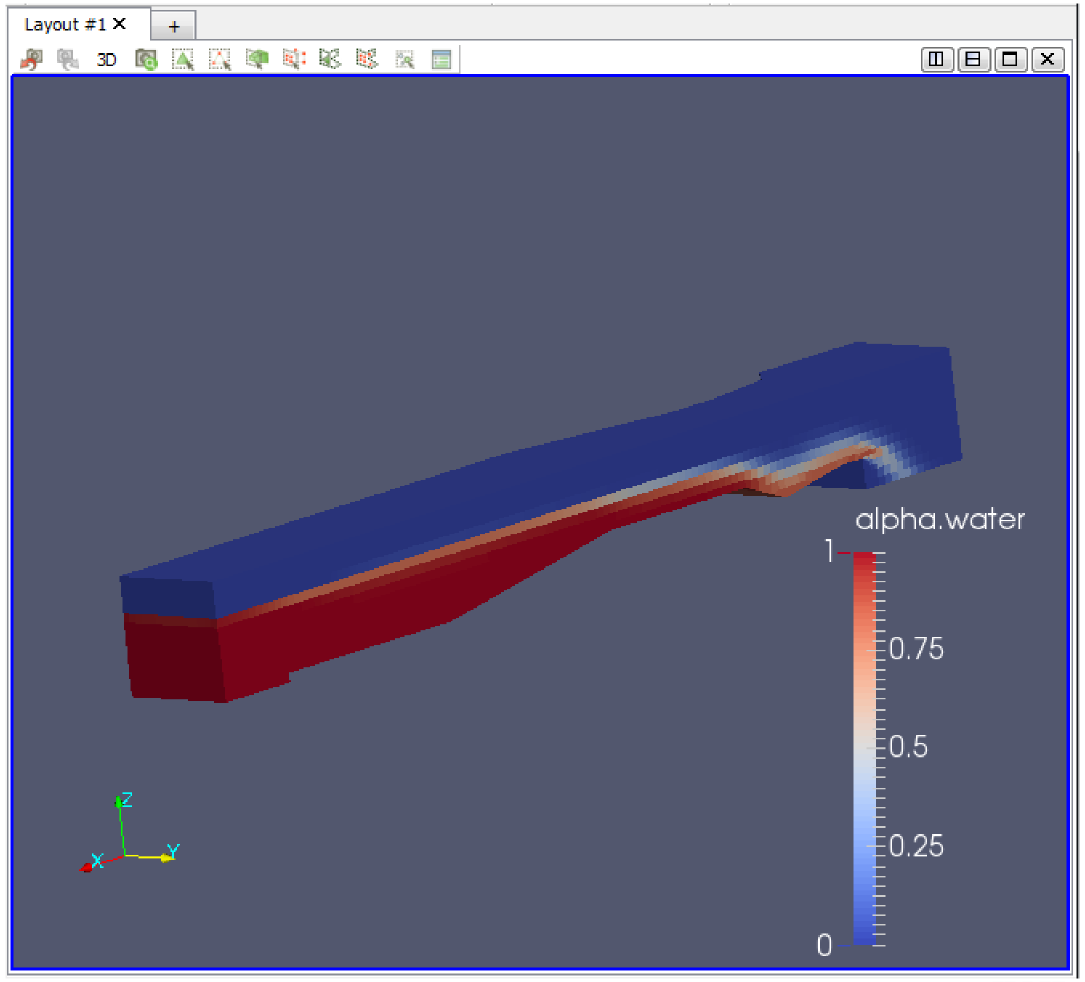
And this is an example on how to change between fields to be rendered.



Post-processing (3/3)

By choosing "`alpha.water`" and turning on the legend, we can see the result below:





The legend can be moved with the mouse.

Thank you!

Any questions?

Virtual Tracer Tests: Coupling CFD and CREng to Simulate WRRFs Unit Processes

Getting Started with OpenFOAM

nelson.marques@fsdynamics.pt; *bruno.santos@fsdynamics.pt*

1st September 2019

Watermatex 2019
DTU
Danmarks Tekniske Universitet
Lund University
IWA the international water association
1 – 4 September 2019 | Copenhagen - Denmark
WATERMATEX 2019
10th IWA Symposium on Modelling and Integrated Assessment
Photo: Jacob Schjørring and Simone Lau, Copenhagen Media Center

FS DYNAMICS optimises your technology

Watermatex 2019 the international water association

Pre-processing

Meshing

Section Contents

Available Meshers (1/4)

Meshers available from within OpenFOAM:

- **blockMesh**: openfoamwiki.net/index.php/BlockMesh
- **snappyHexMesh**: openfoamwiki.net/index.php/SnappyHexMesh
- **foamyHexMesh**: openfoam.org/release/2-3-0/foamyhexmesh/
- **foamyQuadMesh**: 2D version of **foamyHexMesh**.
- **polyDualMesh**: this is mainly a mesh conversion tool, aimed at converting tetrahedral meshes into polyhedral meshes.

Available Meshers (2/4)

Known open-source GUIs for OpenFOAM's meshers:

- **SwiftBlock**: openfoamwiki.net/index.php/Contrib/SwiftBlock
- **SwiftSnap**: openfoamwiki.net/index.php/Contrib/SwiftSnap
- **Helyx-OS**: engys.github.io/HELYX-OS/

Available Meshers (3/4)

Other open-source meshers compatible with OpenFOAM:

- **enGrid**: github.com/enGits/engrid/wiki
- **cfMesh**: sourceforge.net/projects/cfmesh/
- **Netgen**: sourceforge.net/projects/netgen-mesher/
- **TetGen**: wias-berlin.de/software/tetgen/
- **Salome**: www.salome-platform.org
    - Tutorials can be found here: www.caelinux.org/wiki
- **terrainBlockMesher**: github.com/jonasIWES/terrainBlockMesher
- **extBlockMesh**: www.etudes-ng.net/home/development/extBlockMesh

Available Meshers (4/4)

Commercial meshers compatible with OpenFOAM:

- **ANSA**: www.beta-cae.com/ansa.htm
- **Pointwise**: www.pointwise.com
- And most of the ones presented here: openfoamwiki.net/index.php/GUI

Commercial meshers, not 100% compatible:

- Fluent and related ANSYS software, such as T-Grid
- Star-CD and Star-CCM+ by CD-adapco

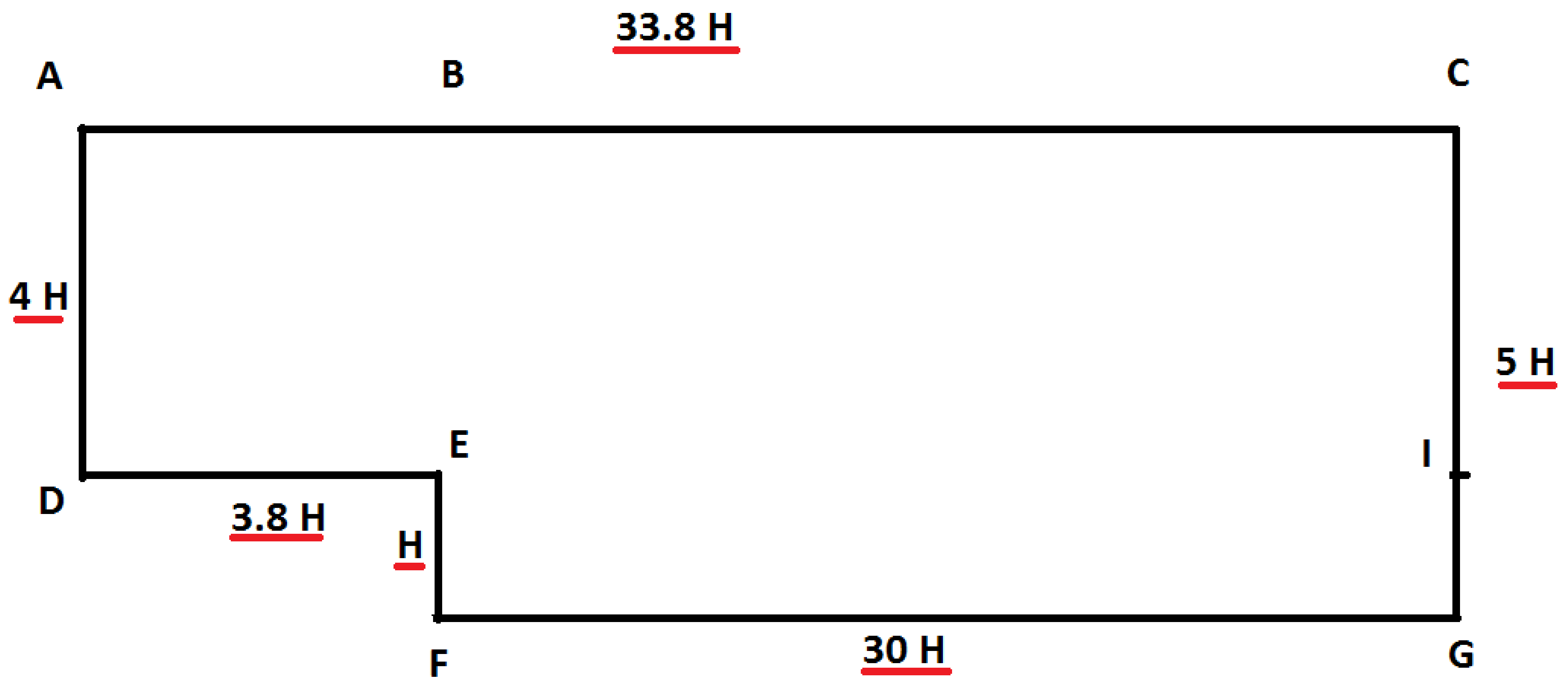
blockMesh (1/13)

Designed for creating a mesh from various sets of blocks, this mesher is very powerful but can quickly get very complicated to use.

We will showcase its basic use with a simple backward-facing step:

- Length of initial channel section: **3.8 H**
- Height of the initial channel section: **4 H**
- Length of the final channel: **30 H**
    - Total length: **33.8 H**
- Height of the step: **H**
    - Which implies that the height of the final channel section is: **5 H**
- The characteristic height **H** to be used will be **1 meter**.

blockMesh (2/13)

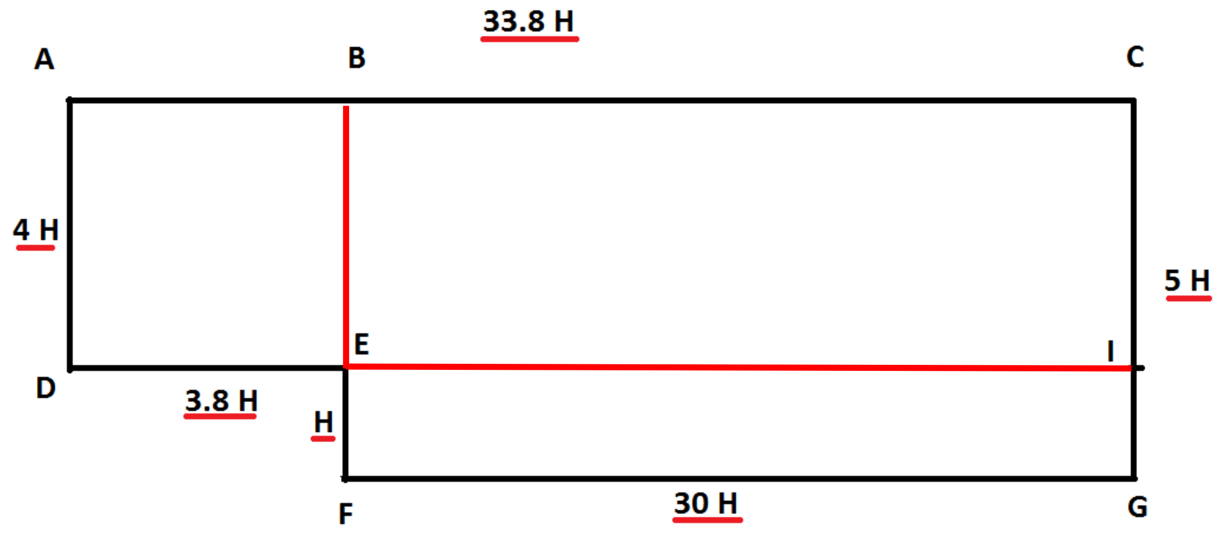


blockMesh (3/13)

Why all those reference points?

Because we will create 3 blocks, namely:

- ADEB
- EFGI
- BEIC

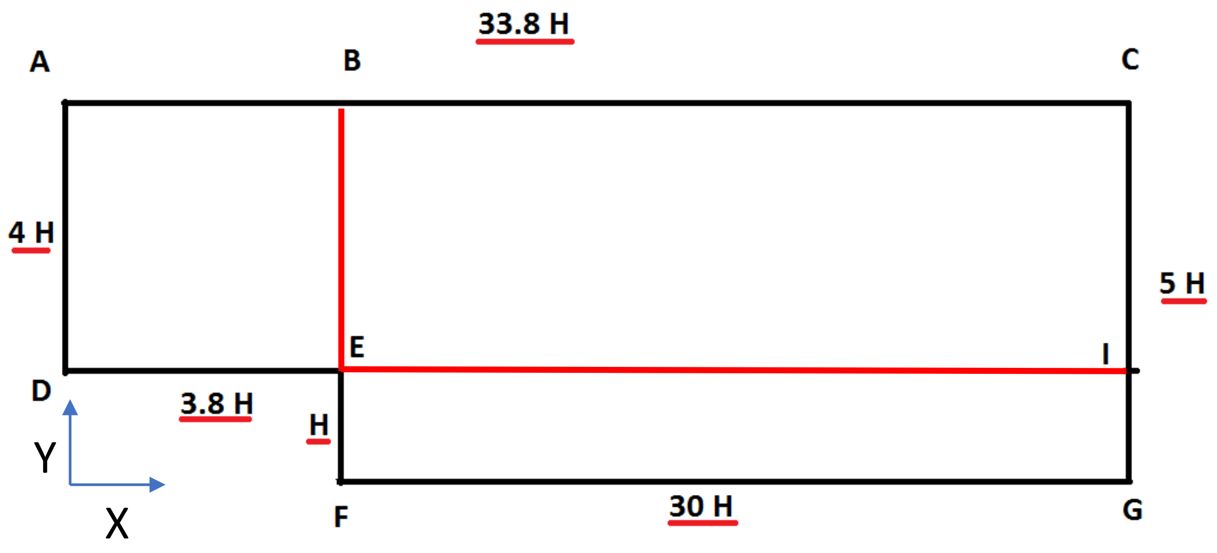


Even though the geometry is defined solely in 2D, OpenFOAM needs the 3rd Dimension just the same, which means that the list of points will be doubled, one for the front, another for the back.

blockMesh (4/13)

To make it easier to create these blocks, we use a few strategies:

- Define the reference X positions for: AD, BEF and CIG
- Define the reference Y positions for: ABC, DEI and FG
- A list of the indexes associated to each point in the front, as well as a list of the indexes for the points in the back.

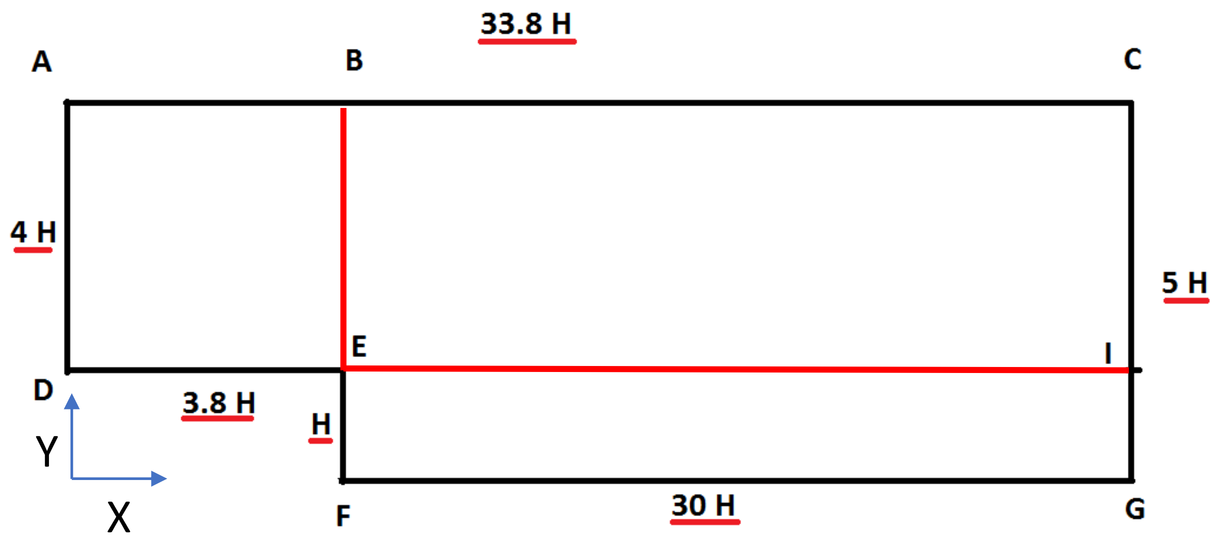


blockMesh (5/13)

In practice, our “blockMeshDict” will look like this:

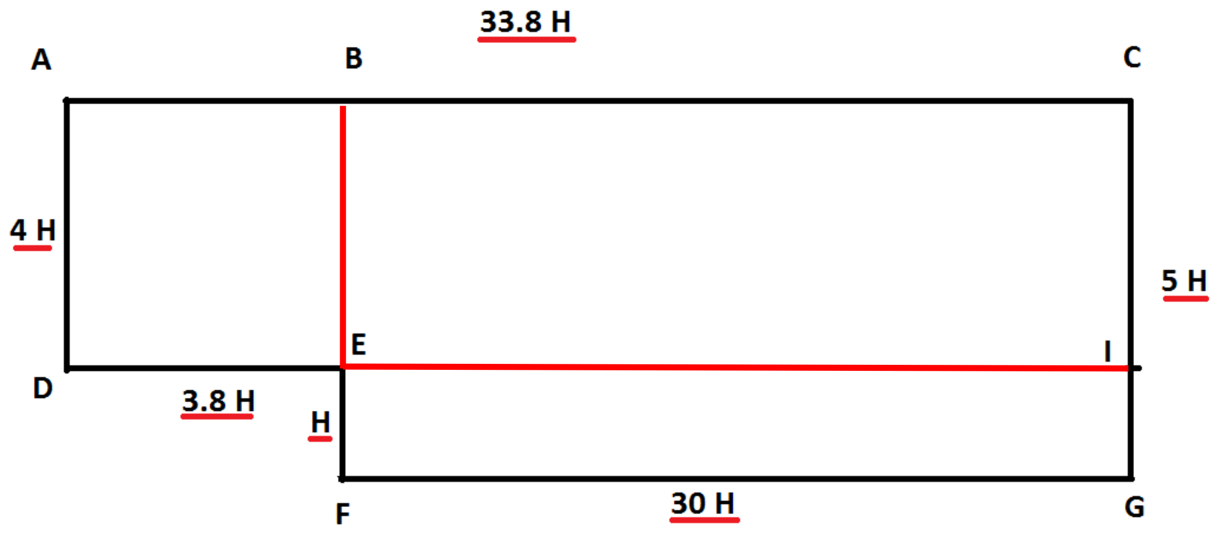
```
convertToMeters 1.0;

// positions ABCDEFGI
ADx       0.0;
BEFx      3.8;
CIGx      33.8;

ABCy    5.0;
DEIy    1.0;
FGy     0.0;

Aa   0;
Ba   1;
Ca   2;
Da   3;
Ea   4;
…
```

```
…
Fa   5;
Ga   6;
Ia   7;

Ab   8;
Bb   9;
Cb   10;
Db   11;
Eb   12;
Fb   13;
Gb   14;
Ib   15;
```



blockMesh (6/13)

List of vertices (front & back):



```
vertices
(
    //Z=0
    ($ADx   $ABCy  0.0)   //A, 0
    ($BEFx  $ABCy  0.0)   //B, 1
    ($CIGx  $ABCy  0.0)   //C, 2
    ($ADx   $DEIy  0.0)   //D, 3
    ($BEFx  $DEIy  0.0)   //E, 4
    ($BEFx  $FGy   0.0)   //F, 5
    ($CIGx  $FGy   0.0)   //G, 6
    ($CIGx  $DEIy  0.0)   //I, 7

…
```

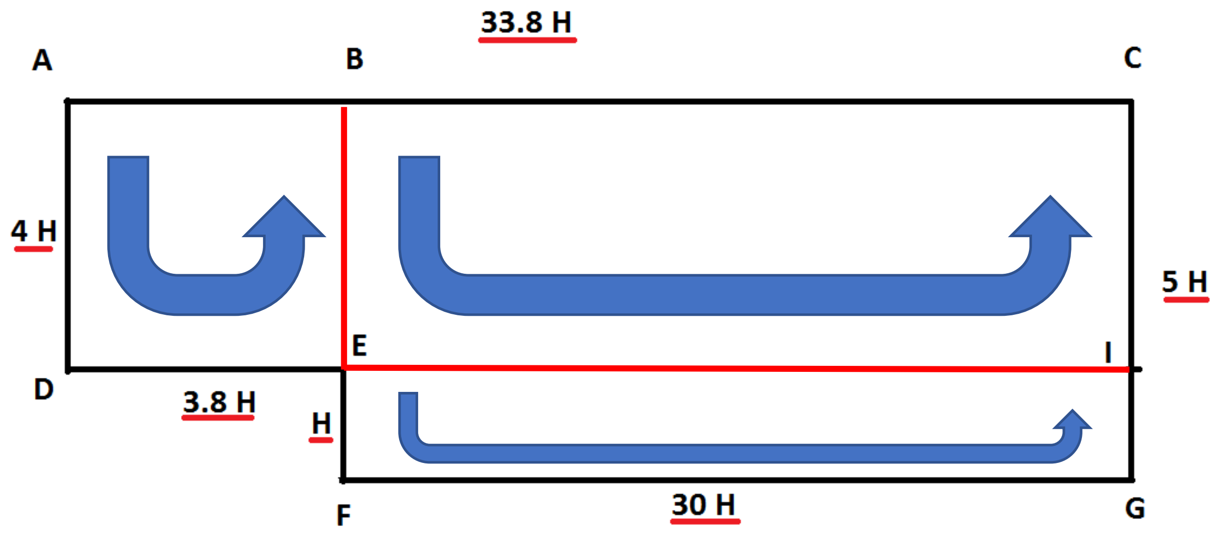
```
…

 //Z=0.1
    ($ADx   $ABCy  0.1)   //A, 8
    ($BEFx  $ABCy  0.1)   //B, 9
    ($CIGx  $ABCy  0.1)   //C, 10
    ($ADx   $DEIy  0.1)   //D, 11
    ($BEFx  $DEIy  0.1)   //E, 12
    ($BEFx  $FGy   0.1)   //F, 13
    ($CIGx  $FGy   0.1)   //G, 14
    ($CIGx  $DEIy  0.1)   //I, 15
);
```

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

blockMesh (7/13)

```
blocks
(
    //ADEB
    hex ($Aa $Da $Ea $Ba $Ab $Db $Eb $Bb) (1 1 1) simpleGrading (1 1 1)

    //EFGI
    hex ($Ea $Fa $Ga $Ia $Eb $Fb $Gb $Ib) (1 1 1) simpleGrading (1 1 1)

    //BEIC
    hex ($Ba $Ea $Ia $Ca $Bb $Eb $Ib $Cb) (1 1 1) simpleGrading (1 1 1)
);
```



FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

blockMesh (8/13)

Geometrical boundaries (1/2):

```
boundary
(
  inlet
  {
    type patch;
    faces
    (
      ($Aa $Da $Db $Ab)
    );
  }

  outlet
  {
    type patch;
    faces
    (
      ($Ca $Ia $Ib $Cb)
      ($Ia $Ga $Gb $Ib)
    );
  }
```

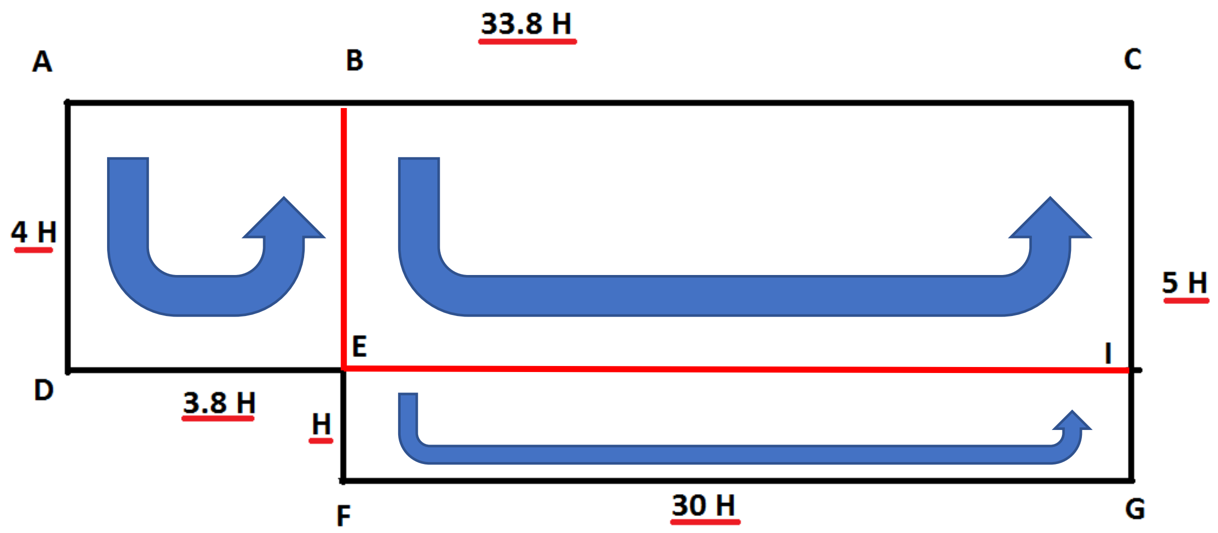
```
  upperWall
  {
    type wall;
    faces
    (
      ($Aa $Ba $Bb $Ab)
      ($Ba $Ca $Cb $Bb)
    );
  }

  lowerWall
  {
    type wall;
    faces
    (
      ($Da $Ea $Eb $Db)
      ($Ea $Fa $Fb $Eb)
      ($Fa $Ga $Gb $Fb)
    );
  }
```

FS DYNAMICS optimises your technology

blockMesh (9/13)

Geometrical boundaries (2/2):

Reminder: the vertices should be defined counter-clockwise and in the same order for the front and back.



```
      frontAndBack
      {
        type empty;
        faces
        (
          ($Aa $Da $Ea $Ba)
          ($Ba $Ea $Ia $Ca)
          ($Ea $Fa $Ga $Ia)

          ($Ab $Db $Eb $Bb)
          ($Bb $Eb $Ib $Cb)
          ($Eb $Fb $Gb $Ib)
        );
      }
);
```

blockMesh (10/13)

Last but not least, the “edges” list and “mergePatchPairs”:

```
edges
(
);

mergePatchPairs
(
);
```

Where:

- **edges**: for providing a list of edge modifiers, e.g.:

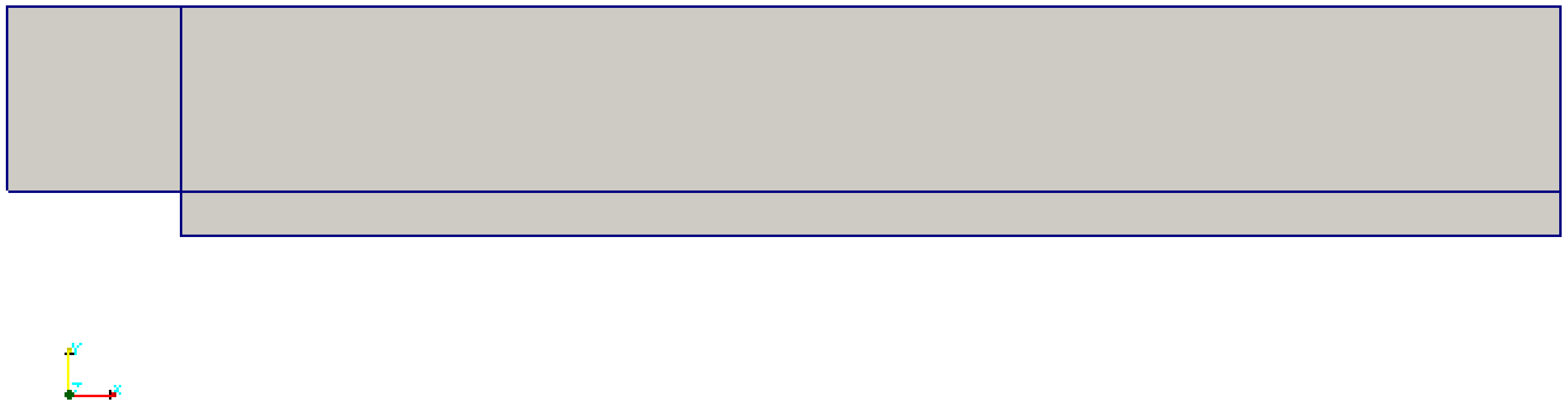
  ```
  arc 0 5 (0.469846 0.17101 -0.5)
  ```

- **mergePatchPairs**: for merging patches, e.g. if we had two geometrical boundaries that we wanted to *stitch together*.

blockMesh (11/13)

Workflow:

1. We use a tutorial case as a basis, for example "`basic/potentialFoam/pitzDaily`".
2. Modify the file "`system/blockMeshDict`".
3. Run **blockMesh**.
4. If all goes well, run **paraFoam**.
5. What we will see in ParaView is something like this:

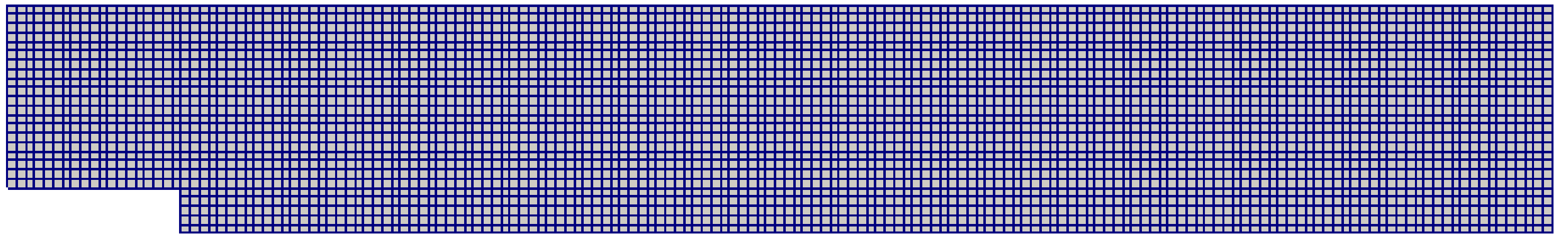


blockMesh (12/13)

Edit the “blockMeshDict” and changing the block list to this:

```
blocks
(
  //ADEB
  hex ($Aa $Da $Ea $Ba $Ab $Db $Eb $Bb) (20 19 1) simpleGrading (1 1 1)

  //EFGI
  hex ($Ea $Fa $Ga $Ia $Eb $Fb $Gb $Ib) (5 150 1) simpleGrading (1 1 1)

  //BEIC
  hex ($Ba $Ea $Ia $Ca $Bb $Eb $Ib $Cb) (20 150 1) simpleGrading (1 1 1)
);
```

Will result in this:



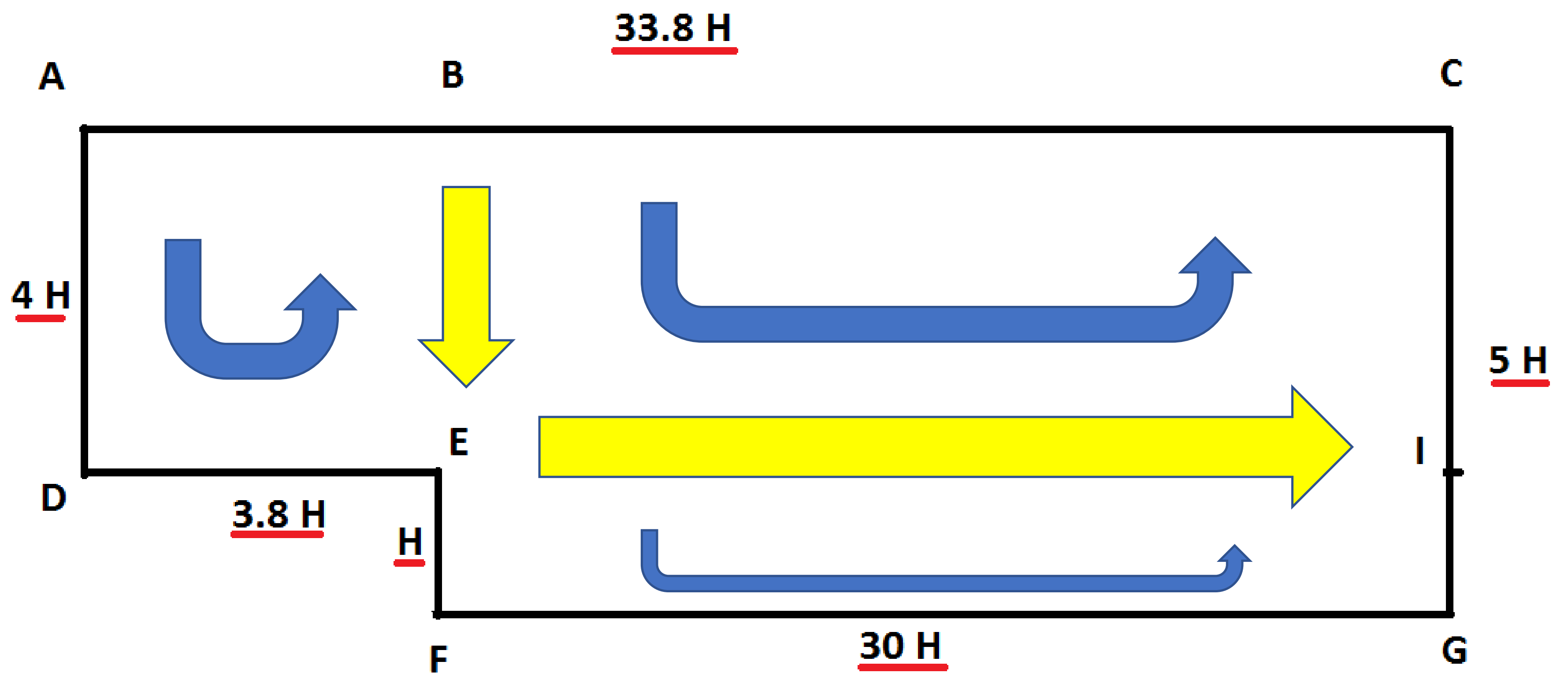
blockMesh (13/13)

The grading over each direction depends in the order of the vertices:

```
hex ($Ba $Ea $Ia $Ca $Bb $Eb $Ib $Cb) (20 150 1)…
```

B → E

E → I



snappyHexMesh (1/49)

This mesher acts as a *chiseller*, given it will work on an initial mesh:

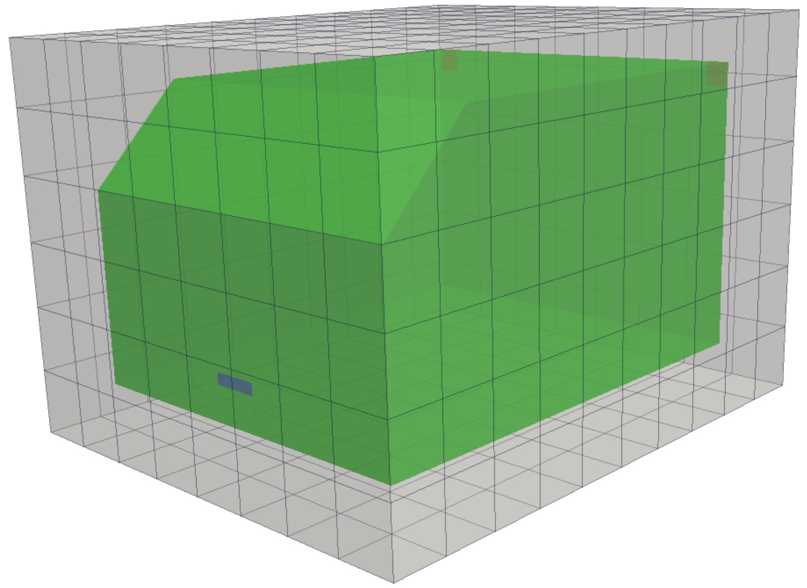
1. Castellation:
   1. Refines the selected edges, surfaces and volumes. All of the selected cells with be divided evenly.
   2. Remove unwanted cells, namely the ones inside or outside of the provided geometrical models.
2. Snapping:
   - The cells near the surfaces of the geometrical models will be cut and/or snapped onto those surfaces.
3. Layer addition:
   - The internal mesh near the surfaces is compacted and prismatic cells are added between the internal mesh and the surfaces.

FS DYNAMICS optimises your technology

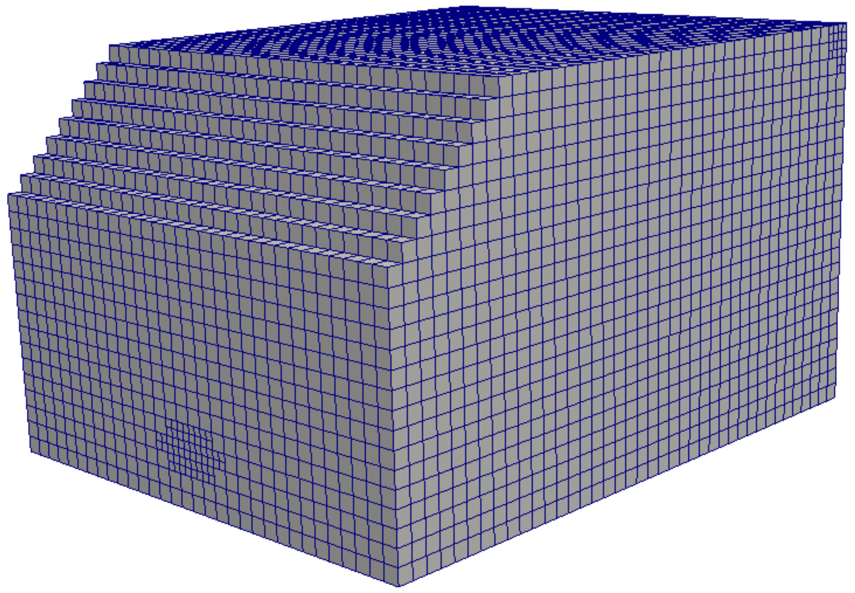
Watermatex 2019 IWA the international water association

snappyHexMesh (2/49)

Examples of each stage (1/2):



Initial (background) mesh



Castellated mesh

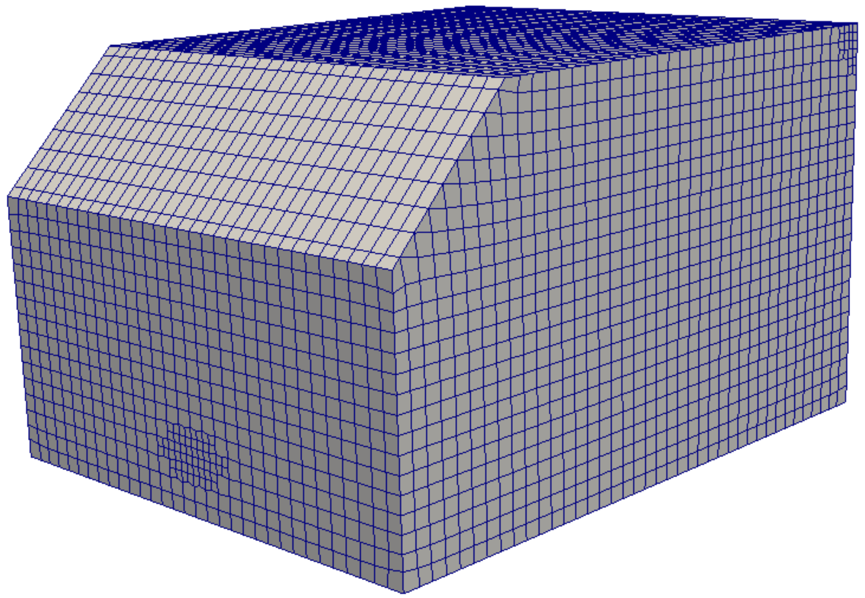
FS DYNAMICS optimises your technology

Watermatex 2019

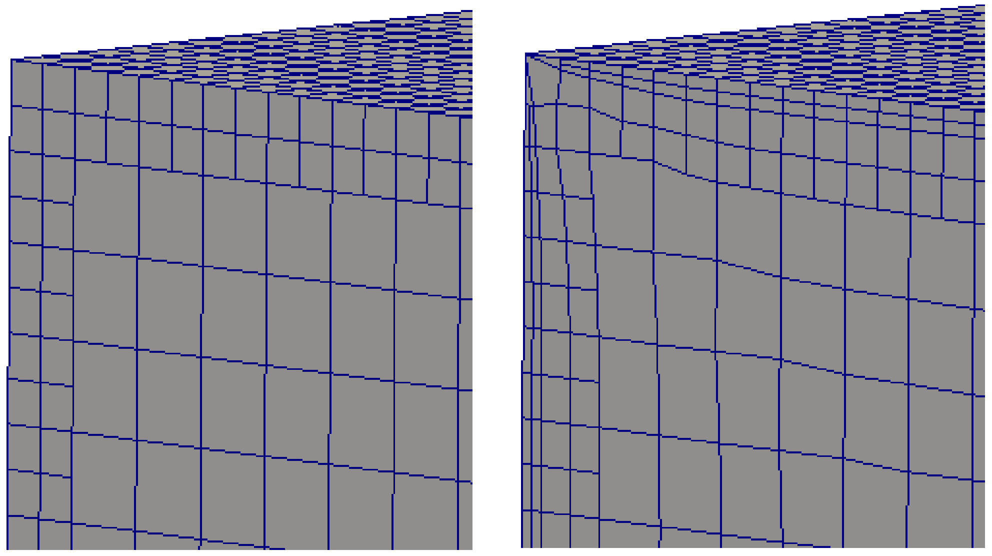
IWA the international water association

snappyHexMesh (3/49)

Examples of each stage (2/2):



Snapped mesh



Before and after layer addition (*section-cut* view)

optimises your technology

snappyHexMesh (4/49)

Surface preparation and import (1/6):

1. Fix the geometry before exporting:
    1. Keep it simple
    2. Remember which side you will mesh
    3. If it is too complex, try something simpler first
    4. Know your physics and solvers
    5. Know the format you are exporting to
    6. Know your units (preferably SI)
    7. Know the location of your geometry
2. Export the geometry to a suitable format
3. Check the resulting discretized geometry

snappyHexMesh (5/49)

Surface preparation and import (2/6):

- OpenFOAM mostly deals with geometric models in STL and Wavefront OBJ files, where both must already be tessellated and preferably in ASCII format.
- Both file formats can handle the identification of separate groups of triangles as patches. For example, in STL, this means that there can be several *solids* in a single STL file.

snappyHexMesh (6/49)

Surface preparation and import (3/6):

Example of an STL:

```
solid backWall
 facet normal 1 0 0
  outer loop
   vertex 5.75 0.687 -0.65
   vertex 5.75 -0.667 -0.188
   vertex 5.75 -0.669 -0.257
  endloop
 endfacet
…
endsolid backWall

solid inlet
…
endsolid inlet
```

Example of an OBJ:

```
# Wavefront OBJ file
# Regions:
#     0    frt-fairing_001_1
#     1    windshield_002_2
…
# points    : 132871
# triangles : 331653
…
v -0.00568945 0.0242072 1.6e-08
…
v 1.74458 -0.00375868 1.01589
g frt-fairing_001_1
f 64491 64463 65119
…
f 17054 16748 17078
g windshield_002_2
f 66424 67098 66986
…
```

snappyHexMesh (7/49)

Surface preparation and import (4/6):

Once the file is in STL or OBJ format, we can check how OpenFOAM will interpret, by running:

```
surfaceCheck path/to/the_file.stl
```

It will give us a lot of information, such as:

- Bounding box
- Regions (= solids)
- If any illegal triangles were found
- An histogram of the quality of the triangles
- Lengths of the triangle edges and nearness of vertices
- If the geometry is closed and identifies parts if not closed

snappyHexMesh (8/49)

Surface preparation and import (5/6):
Place the exported geometry file in the folder “`constant/triSurface`”.

Move/translate the whole geometry so that the center of the geometry coincides with the origin of the referential.

This can be done with **surfaceTransformPoints**, e.g.:

```
surfaceTransformPoints -translate '(-4.25 0.687 -0.55)' \
      constant/triSurface/halfParshall.org.stl \
      constant/triSurface/halfParshall.stl
```

snappyHexMesh (9/49)

Surface preparation and import (6/6):
Configure the dictionary file "`system/surfaceFeatureExtractDict`", for extracting feature edges from the geometry, e.g.:

```
"halfParshall.stl"
{
    extractionMethod    extractFromSurface;
    extractFromSurfaceCoeffs
    {
        includedAngle   150;
    }
    writeFeatureEdgeMesh    no;
}
```

Then simply run:

```
surfaceFeatureExtract
```

optimises your technology

Watermatex 2019

the international water association

snappyHexMesh (10/49)

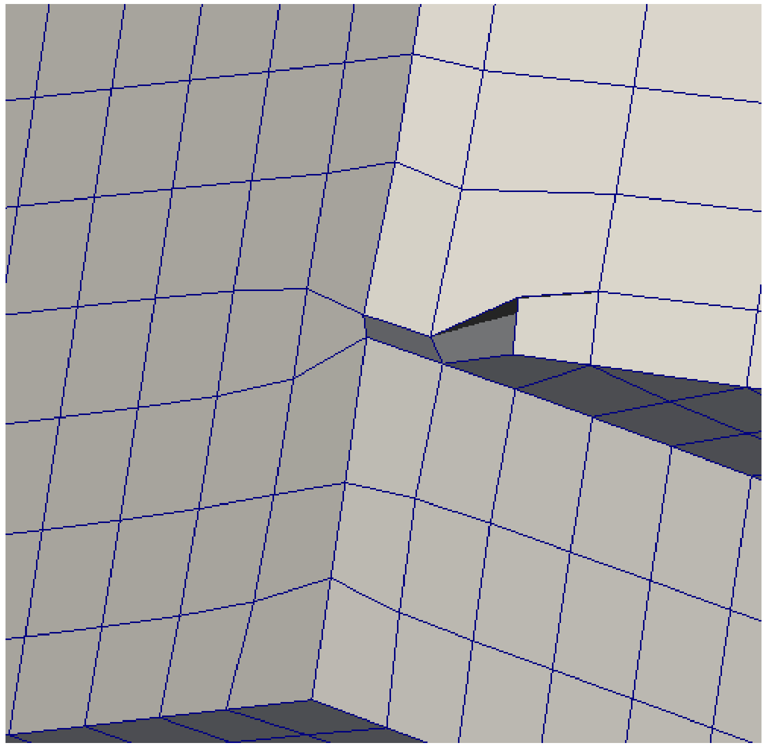
Background mesh (1/6):

Also known as “initial mesh” or “base mesh”, this is the initial mesh that will be used by **snappyHexMesh**.

Which means that:

- The initial mesh should be similar to the geometry;
- Or at least good reference points from the geometry must be part of this initial mesh.

Otherwise, things like this happen.

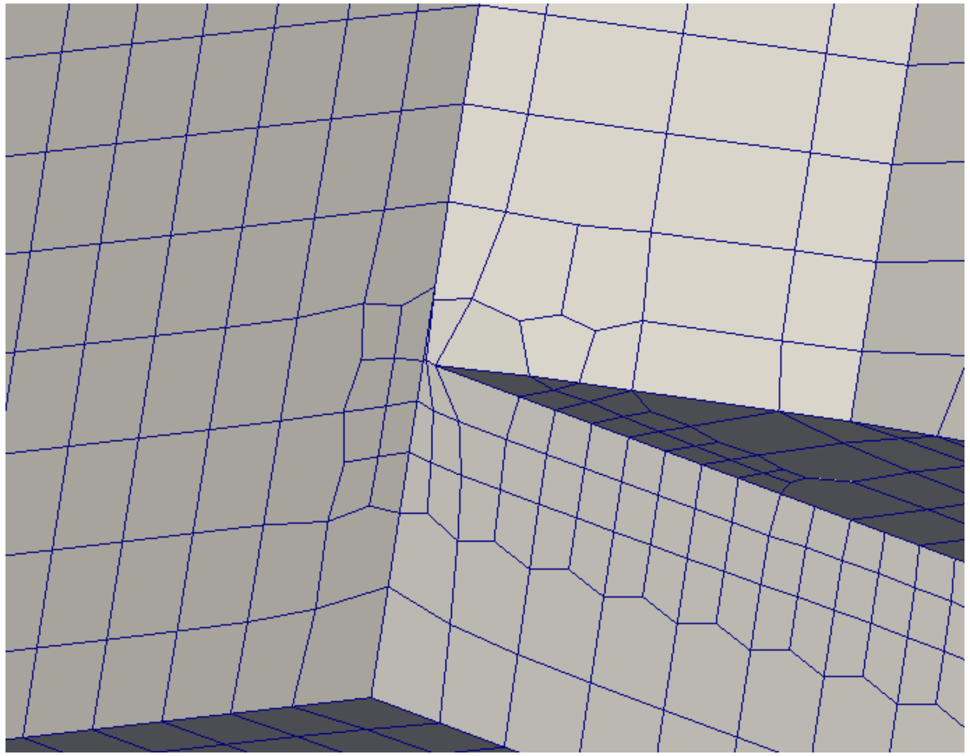


The image shown is for level 0, i.e. initial mesh.

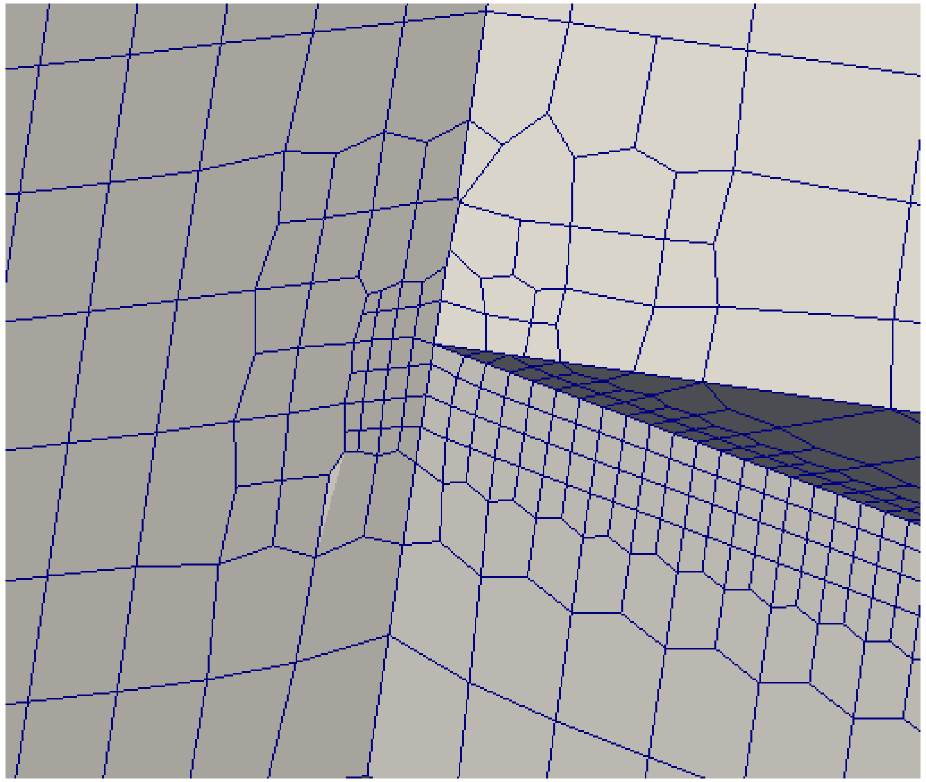
snappyHexMesh (11/49)

Background mesh (2/6):

Although we can try to fix it with increasing the refinement in the castellation step with **snappyHexMesh**:



Level 1 refinement



Level 2 refinement

snappyHexMesh (12/49)

Background mesh (3/6):

The usual is a single block done with **blockMesh**, e.g.:

```
convertToMeters 1;

vertices
(
    (-5.75 -0.686883 -0.65)
    (5.75  -0.686883 -0.65)
    (5.75   0.687 -0.65)
    (-5.75  0.687 -0.65)
    (-5.75 -0.686883 0.65)
    (5.75  -0.686883 0.65)
    (5.75   0.687 0.65)
    (-5.75  0.687 0.65)
);
```

```
blocks
(
    hex (0 1 2 3 4 5 6 7)
    (115 14 13)
    simpleGrading (1 1 1)
);
```

```
patches
(
    patch maxX
    ( (1 2 6 5) )
    patch minX
    ( (0 4 7 3) )
    patch maxY
    ( (3 7 6 2) )
    patch minY
    ( (1 5 4 0) )
    patch maxZ
    ( (4 5 6 7) )
    patch minZ
    ( (0 3 2 1) )
);
```

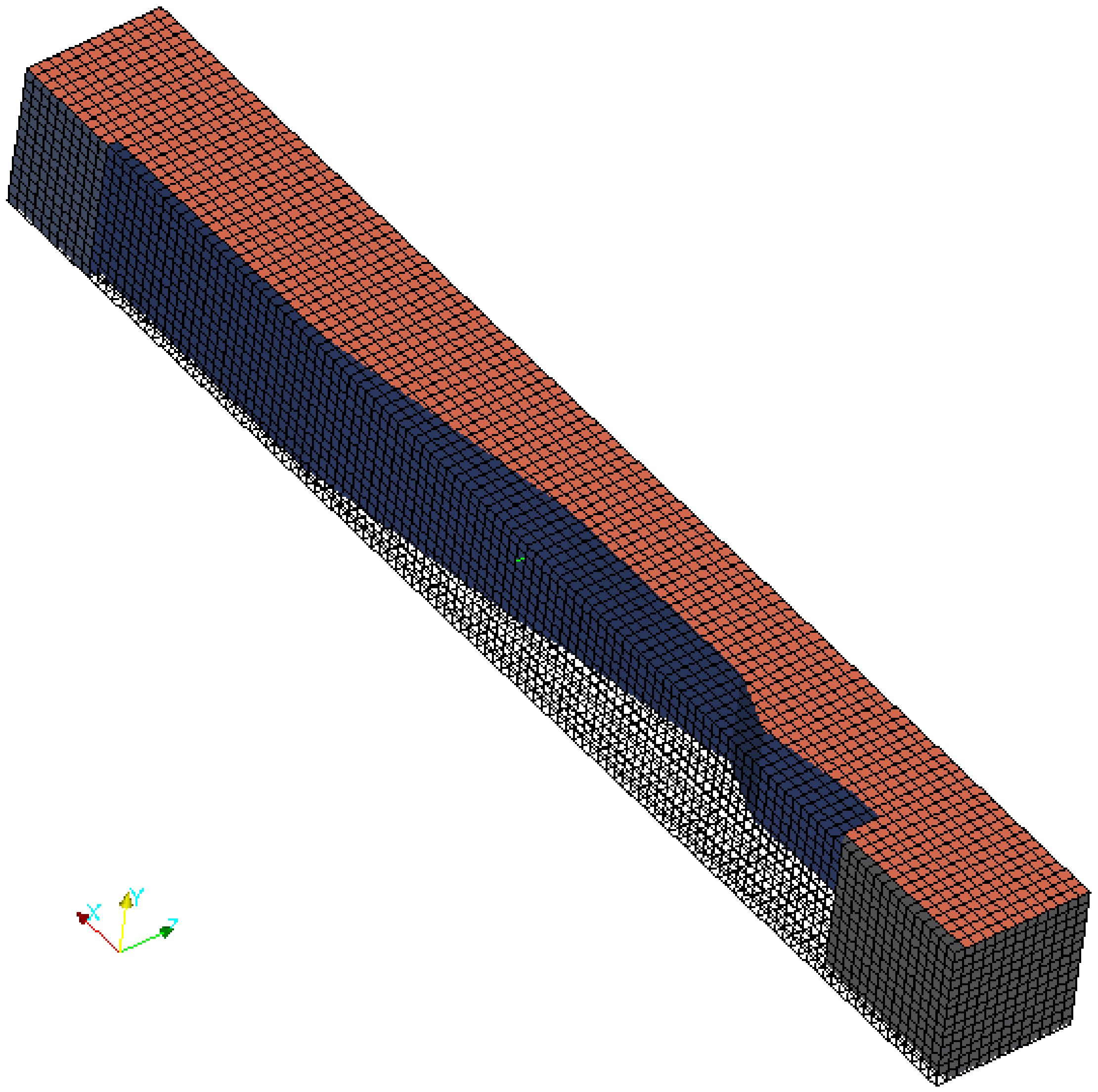
snappyHexMesh (13/49)

Background mesh (4/6):

Here is the result in our example:

As shown, this is a very good initial mesh for our model given the several reference points that are present in this initial mesh.

The detail is that we need an additional layer outside of this mesh.



FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

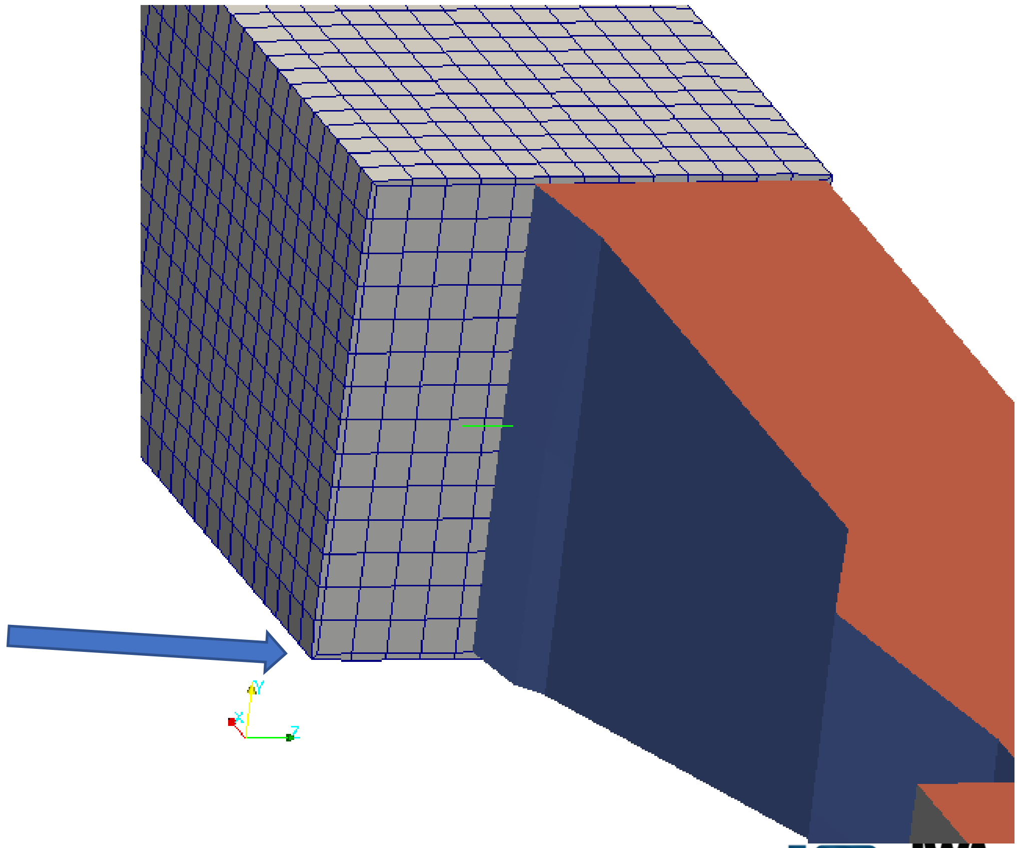
snappyHexMesh (14/49)

Background mesh (5/6):

The additional layer outside of the mesh shown is needed for ensuring that **snappyHexMesh** is able to properly correlate this initial mesh with the geometrical model.

One way is to *do the math* and extend the vertices defined in "blockMeshDict" file, along with 2 more cells on all direction.

The other is to rely on **extrudeMesh** to do this for us.



snappyHexMesh (15/49)

Background mesh (6/6):

Example file “`system/extrudeMeshDict`” from our case:

```
constructFrom mesh;

sourceCase ".";
sourcePatches(maxX  minX  maxY  minY  maxZ  minZ);

extrudeModel            linearNormal;
nLayers                 1;
expansionRatio          1.0;

linearNormalCoeffs
{
    thickness           0.02;
}

mergeFaces false;
mergeTol 0;
```

snappyHexMesh (16/49)

Mesh parameters (1/32):

The file "`system/snappyHexMeshDict`" has all of the necessary settings for **snappyHexMesh** to manipulate the background mesh.

The main reference file is present in the application's source code folder, whose location is shown with this command:

```
echo $FOAM_UTILITIES/mesh/generation/snappyHexMesh
```

Other examples can be found with this command:

```
find $FOAM_TUTORIALS -name snappyHexMeshDict
```

snappyHexMesh (17/49)

Mesh parameters (2/32):

The structure of the file is as follows:

- Initial parameters which control what steps to perform.
- "`geometry`" block, where we list the geometrical entities we want to either mesh onto or use as refinement references.
- "`castellatedMeshControls`" block, for the castellation step.
- "`snapControls`" block, for the snapping step.
- "`addLayersControls`" block, for the layer adding step.
- "`meshQualityControls`" block, for quality control parameters that are used during the snapping and layer adding steps.
- Last parameters are for debugging and point tolerance.

snappyHexMesh (18/49)

Mesh parameters (3/32) – run steps selection:
There are only 3 options for this section of the dictionary file:

```
castellatedMesh true;
snap            true;
addLayers       false;
```

Setting each one to true or false will tell **snappyHexMesh** to proceed with each step.

Keep in mind that each one of these 3 steps has associated a block of settings for each, as listed in the previous slide.

snappyHexMesh (19/49)

Mesh parameters (4/32) – Geometry definitions (1/5):

The geometry definitions are defined within this block:

```
geometry
{
   //...
};
```

Inside this block the user should add as many geometry objects as needed, where each block is identified as follows:

```
object_file_name.extension
{
    type the_type_of_object;

    //settings for this object
}
```

snappyHexMesh (20/49)

Mesh parameters (5/32) – Geometry definitions (2/5):

From our example case, we have two geometries:

```
"halfParshall.stl"
{
  type triSurfaceMesh;
  regions
  {
    backWall
    {
      name backWall;
    }
…
    top
    {
      name top;
    }
  }
}
```

```
refinementBox
{
  type    searchableBox;
  min     (-4.35 -1 -0.30);
  max     (-4.20  1 -0.15);
}
```

optimises your technology

snappyHexMesh (21/49)

Mesh parameters (6/32) – Geometry definitions (3/5):

The details are as follows:

- Type "`triSurfaceMesh`" is the one used for external model files.
  - As the designation implies, this model must be in a file format that uses a triangle discretization of the surfaces.
  - We're using STL, as it's the easiest one to generate and manipulate.

(continues on next slide…)

snappyHexMesh (22/49)

Mesh parameters (7/32) – Geometry definitions (4/5):

- The regions block is where we can rename the solids provided in the STL and give them the names we want.
  - These renamed names will later be used for defining the patches that make up the surface mesh of our final mesh.
  - This is also used because without this renaming step, the default names assigned by **snappyHexMesh** would likely be something like this:

```
halfParshall_backWall
halfParshall_bottomWall
halfParshall_sideWall
```

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

snappyHexMesh (23/49)

Mesh parameters (8/32) – Geometry definitions (5/5):

Other internal geometrical entities can be used, some examples:

- `searchableBox` – defines a box by bounding points (which we used in our example case);
- `searchableSphere` – defines a sphere by center and radius;
- `searchableCylinder` – defines a cylinder by height vector and radius;
- `searchablePlate` – defines a plate by origin and span;
- `searchablePlane (planeType PointAndNormal)` – defines a plane by point and normal vector;
- `searchablePlane (planeType 3Points)` – defines a box by plane by three points.

FS DYNAMICS optimises your technology

Watermatex 2019 the international water association

snappyHexMesh (24/49)

Mesh parameters (9/32) – Castellation controls (1/11):

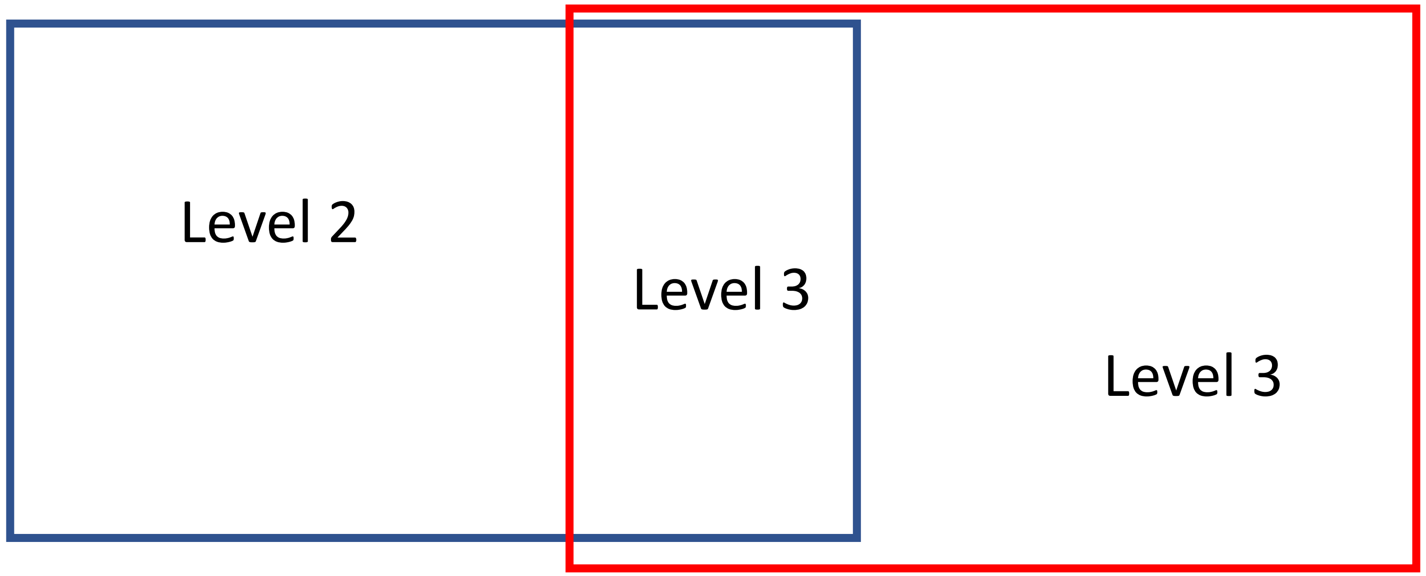
First we need to understand how this step handles refinement levels:

1. Initial mesh is the level of refinement 0 (zero).
2. Each level of refinement indicates the multiple of 2 for dividing cells. In other words:
   - level 1: a cell from the initial mesh is split into 2 parts on all major directions (X, Y, Z), in it's own referential. In other words, each cell will be subdivided into 8 smaller cells.
   - level 2: cell split into 4 parts over X,Y,Z → will be subdivided into 64 smaller cells.
   - level 3: split into 8 parts over X,Y,Z → 512 smaller cells.

snappyHexMesh (25/49)

Mesh parameters (10/32) – Castellation controls (2/11):

3. Refinement levels are not cumulative:
   - if two or more overlapping zones and/or surfaces are set to different levels of resolution, it's only the greatest value that will be used.
   - For example:



snappyHexMesh (26/49)

Mesh parameters (11/32) – Castellation controls (3/11):

Choosing which cells to refine is done based on the three major types of geometries in 3D space:

- Lines: Provided as feature edges in OpenFOAM's the file format “.eMesh”.
- Surfaces: More specifically, all of the surfaces from the geometries defined in the “geometry” block.
- Volumes: Can refine inside or outside of *closed shells* from the “geometry” block; or distance based for any geometric entity from the “geometry” block.

snappyHexMesh (27/49)

Mesh parameters (12/32) – Castellation controls (4/11):

In block "`castellatedMeshControls`", we focus on:

- "`maxLocalCells`" is used when running in parallel. This is the guideline on when it should transfer excess cells to other processor sub-domains, assuming the others are less populated.
- "`maxGlobalCells`" is the maximum number of cells that **snappyHexMesh** will allow to be generated in the refinement step. A rule of thumb is that mesh generation can take up somewhere between <u>*1 and 2 GB of RAM for each one million cells*</u>.

snappyHexMesh (28/49)

Mesh parameters (13/32) – Castellation controls (5/11):

- “features” provides a list of nameless blocks that define the feature edge files to be used for refinement and later on for snapping.
  - For each nameless block, we can define the file to be used and the associated refinement level.
  - The advantages of using feature edges as references for the refinement/castellation step depend on the geometry at hand.
  - Best to define the refinement level for each file to be 0; otherwise, there is a risk of having refined cells in very impractical locations.

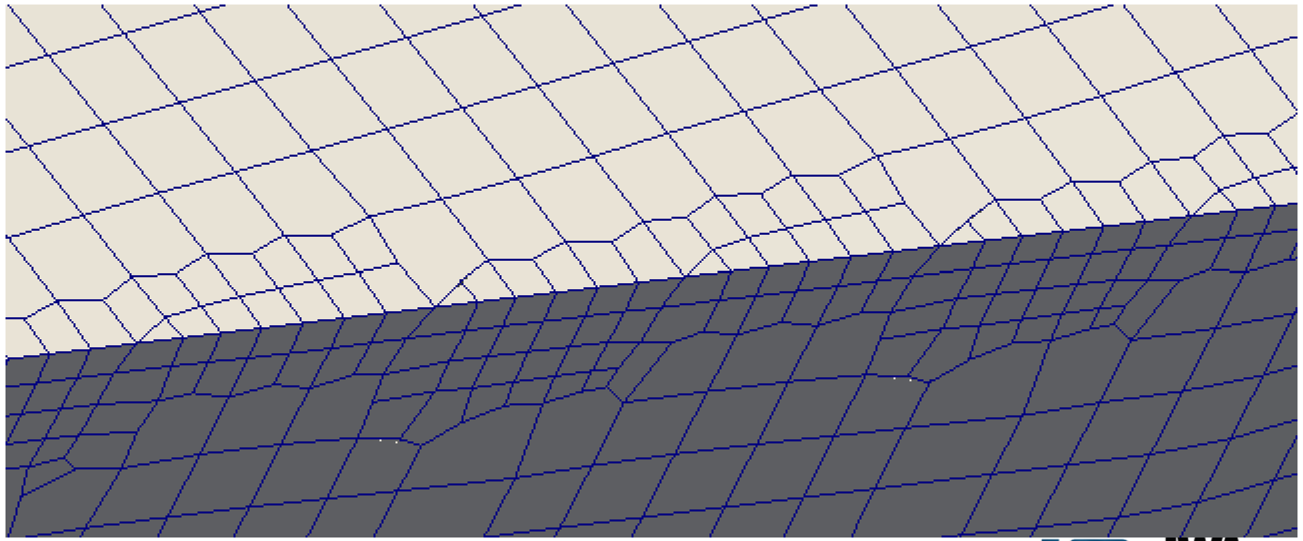
snappyHexMesh (29/49)

Mesh parameters (14/32) – Castellation controls (6/11):

From our example case, this is the “features” block:

```
features
(
  {
    file "halfParshall.eMesh";
    level 0;
  }
);
```

This is what would happen with level 1



snappyHexMesh (30/49)

Mesh parameters (15/32) – Castellation controls (7/11):

- "`refinementSurfaces`" block provides the refinement settings for the surfaces on the geometries defined in the geometry block.
  - The structure to be followed is similar to the one used in the geometry block.
  - For each *solid* (named region) we can define two sets of refinement levels, namely the minimum and the maximum level.
  - **Note**: It is good to remember that a more uniform mesh is usually preferable to a mesh with many mesh level refinement transitions.

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

snappyHexMesh (31/49)

Mesh parameters (16/32) – Castellation controls (8/11):

From the example case – “`refinementSurfaces`” block:

```
refinementSurfaces
{
  "halfParshall.stl"
  {
    level (0 0);
    regions
    {
      backWall
      {
        level (0 0);
      }
      bottomWall
      {
        level (0 0);
      }
      outlet
      {
        level (0 0);
      }
      symmetry
      {
        level (0 0);
      }
      top
      {
        level (0 0);
      }
    }
  }
}
```

snappyHexMesh (32/49)

Mesh parameters (17/32) – Castellation controls (9/11):

- "`resolveFeatureAngle`" is in essence what defines the smallest angle between two surfaces that we can use to consider if there is a sharp edge between the two.
- "`refinementRegions`" already mentioned regarding refinement volumes, which are based on the geometries defined in "geometry".

From our example case:

```
refinementRegions
{
  refinementBox
  {
    mode inside;
    levels ((1e-15 2));
  }
}
```

The first value, "1e-15", is meant to be used only for the "distance" mode

FS DYNAMICS optimises your technology

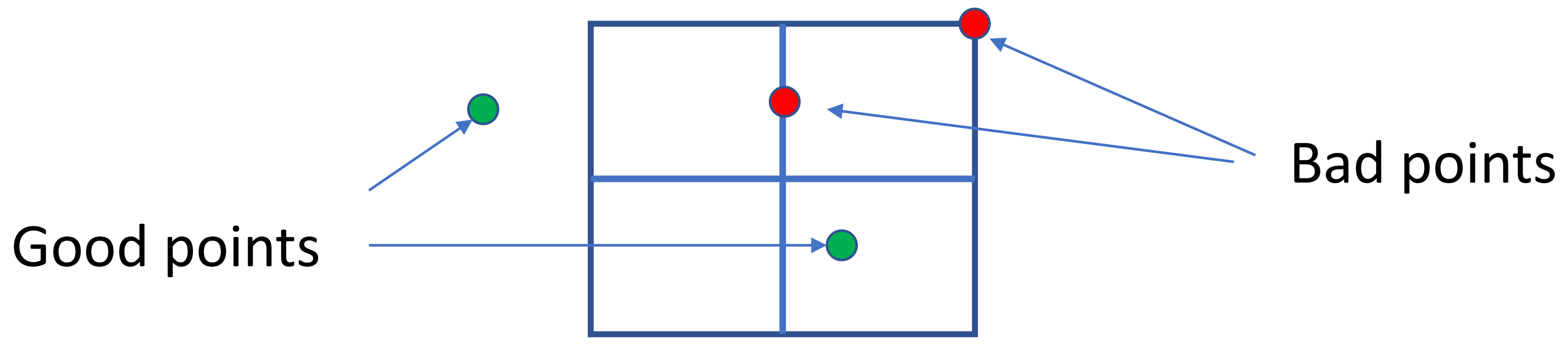
Watermatex 2019

the international water association

snappyHexMesh (33/49)

Mesh parameters (18/32) – Castellation controls (10/11):

- "`locationInMesh`" will tell the mesher that a particular point is inside or outside of the closed surface. Use with care!
- If the point is inside, it will preserve only the mesh inside the geometry.
- If the point is outside, it will preserve only the mesh outside of the geometry.
- Examples:



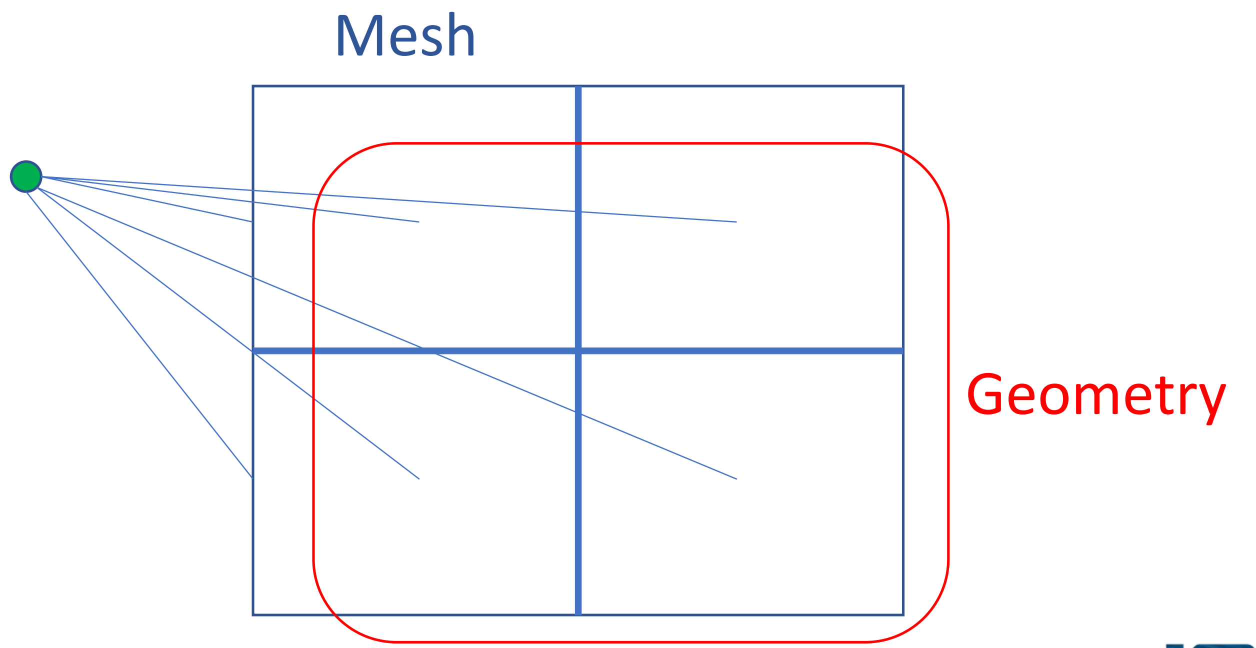
snappyHexMesh (34/49)

Mesh parameters (19/32) – Castellation controls (11/11):

How is “`locationInMesh`” used?

It’s used for drawing lines between this reference point and the centers of cells and faces, to ascertain which cells are inside or outside of the mesh.

Example for when the point outside of the geometry.



snappyHexMesh (35/49)

Mesh parameters (20/32) – Snapping controls (1/5):

The block for this snapping/morphing step starts and ends like this:

```
snapControls
{

//...

}
```

As for the specific parameters for this block, it is all well explained in the comments already available in OpenFOAM's example, therefore we'll address how each parameter can affect the snapping process.

optimises your technology

Watermatex 2019

the international water association

snappyHexMesh (36/49)

Mesh parameters (21/32) – Snapping controls (2/5):

- "`nSmoothPatch`" is the number of iterations for smoothing. When set to 0, stays true to the initial mesh shape; when set too high, will result in a surface mesh that will resemble a seawater blowfish.
- "`tolerance`" is the relative distance for cell edge length for snapping points.
  - Common values are 1.0 and 2.0.
  - If values are too high, it risks snapping vertexes on the mesh that have nothing to do with the nearest surface.
  - If the values are too low, snapping might never occur.

snappyHexMesh (37/49)

Mesh parameters (22/32) – Snapping controls (3/5):

- “`nSolveIter`” is the number of iterations for adjusting the mesh.
  - Set to 0 if the surface of the base mesh is parallel to the surfaces of the final mesh.
  - Other integer values above zero can improve the resulting mesh.
- “`nRelaxIter`” is pretty much a must-use.
  - Default value of 5 usually does a very good job.
  - More iterations can make it slower than needed.

snappyHexMesh (38/49)

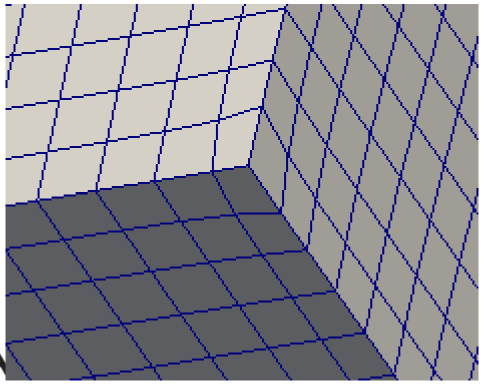
Mesh parameters (23/32) – Snapping controls (4/5):

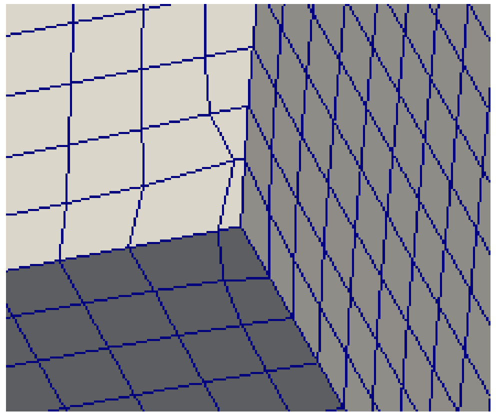
- “`nFeatureSnapIter`” is the number of iterations for snapping to feature edges.
  - The default value of 10 is usually the best value.
  - Too low can result in an incomplete morph.
  - Too high can lead to strange mesh distortions.
- “`implicitFeatureSnap`” when set to true, you don't need the “`.eMesh`” files.
- “`explicitFeatureSnap`” when set to true, it will use the “`.eMesh`” files listed in the features block at “`castellatedMeshControls`”.

snappyHexMesh (39/49)

Mesh parameters (24/32) – Snapping controls (5/5):

- "`multiRegionFeatureSnap`" when set to true, it will only work if "`explicitFeatureSnap`" is also set to true.
  - Only useful for multi-region meshing, namely on both inside and outside.
  - Using this is risky, because it will enforce mesh quality controls for both sides of the mesh, namely inside and outside, which can result in *crooked looking* mesh.


When set to false


When set to true

snappyHexMesh (40/49)

Mesh parameters (25/32) – Layer addition controls (1/7):

The importance of adding layers is related to the requirements needed for the wall treatment models being used for the simulation.

The block for this layer addition step starts and ends like this:

```
addLayersControls
{

//...

}
```

snappyHexMesh (41/49)

Mesh parameters (26/32) – Layer addition controls (2/7):

The more relevant layer adding parameters are:

- "`relativeSizes`" when set to true, will use dimensioning relative to the cell edges where the layers will be added.
- "`layers`" block lists all patches (boundary surfaces) that should have layers added or not to them. It can even allow per-patch definition of the layer adding parameters.
- "`expansionRatio`" is the factor of how each layer relates to the previous on added.
- "`finalLayerThickness`" is the first reference layer size, which is the thickness of the layer farthest from the original surface.

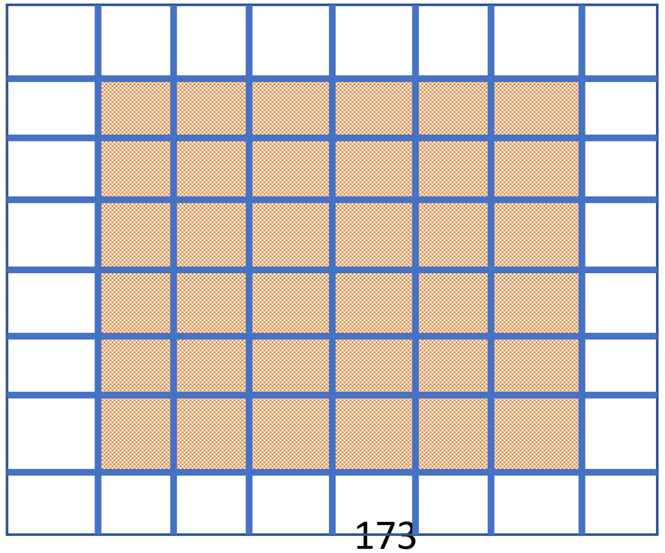
FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

snappyHexMesh (42/49)

Mesh parameters (27/32) – Layer addition controls (3/7):

- "`minThickness`" is the smallest desired thickness at the middle axis of a layer's cell.
- "`nBufferCellsNoExtrude`" is the number of cells on the mesh before adding layers, relative to the border of a patch. For example, with a value of 1, only the centre cells on the mesh below would get layers added:



This is the surface mesh of the patch on which layers will be added.

FS DYNAMICS optimises your technology

snappyHexMesh (43/49)

Mesh parameters (28/32) – Layer addition controls (4/7):

Example of the “layers” block:

```
layers
{
  backWall
  {
    nSurfaceLayers 1;
  }

  bottomWall
  {
    nSurfaceLayers 1;
  }


…
```

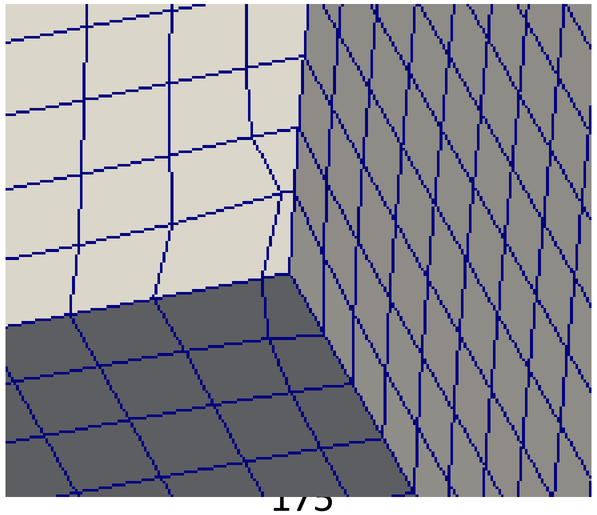
```
…

inlet
{
  nSurfaceLayers 1;
  // Per patch layer data
  expansionRatio       1.0;
  finalLayerThickness 2.0;
  minThickness         0.1;
}

…
```

snappyHexMesh (44/49)

Mesh parameters (29/32) – Layer addition controls (5/7):

Usual sources of problems for not being able to add layers with **snappyHexMesh**:

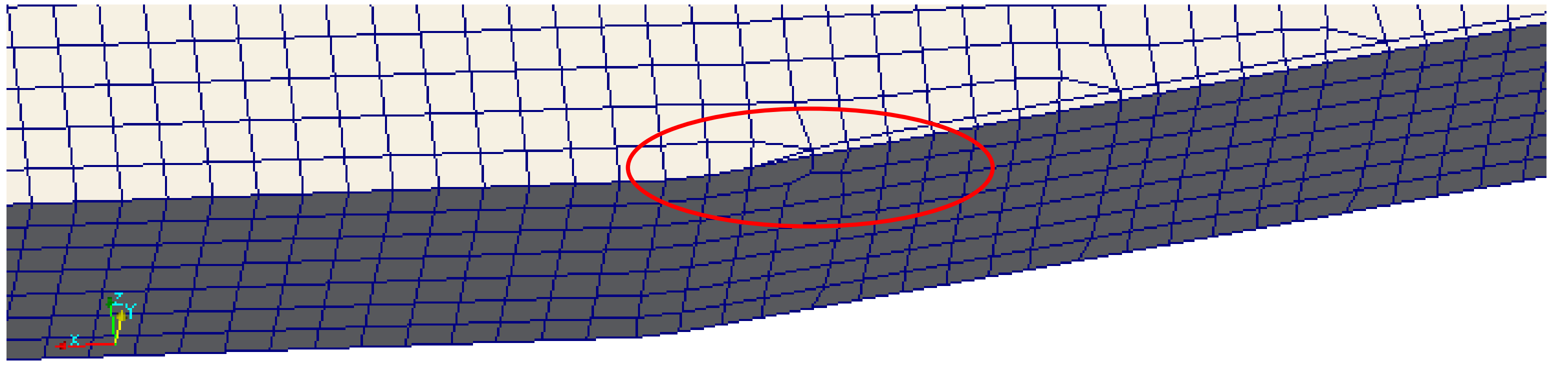
- The mesh resulting from the snapping step has quality issues.
  - For example, the image below demonstrates that layers cannot be added near the shown distorted cells.



snappyHexMesh (45/49)

Mesh parameters (30/32) – Layer addition controls (6/7):

- Complex surfaces identified as the same *solid* in STL, can result in calculations of where the layer should start and where it should end.
  - For example, if a complex profile such as the "`bottomWall`" in our example case, having all surfaces at the bottom catalogued as being part of "`bottomWall`", results in this layer adding flaw:



snappyHexMesh (46/49)

Mesh parameters (31/32) – Layer addition controls (7/7):

- Layer sizes can be relative or absolute.
  - If the mesh resulting from the snapping step gives rise to very small cells where cuts had to be done, then very thin layers can appear where the cells became smaller, when using relative sizing.
  - Absolute sizing can fix this issue, although it requires a clear notion of the geometrical sizes desired for the added layers.
- When layers are set to be too large, it can result in very distorted internal mesh, given that the layers are added by compressing the internal mesh, <u>not</u> by cutting the cells near the surface.

snappyHexMesh (47/49)

Mesh parameters (32/32) :

As for the remaining controls:

- Mesh quality controls are usually well calibrated in the tutorial cases and rarely need to be changed.
    - Nonetheless, changing them should be done with simple small test cases, for diagnosing if they will affect your mesh or not.
- The "`debug`" flag(s) is(are) usually only needed for diagnosing in which exact mesh operation *things went wrong*.
- The "`mergeTolerance`" rarely needs to be changed from the default "1E-6". This is relative to the bounding box of the whole mesh.

snappyHexMesh (48/49)

Visualisation (1/2):

When it comes to visualizing the mesh, the 3 important rules are:

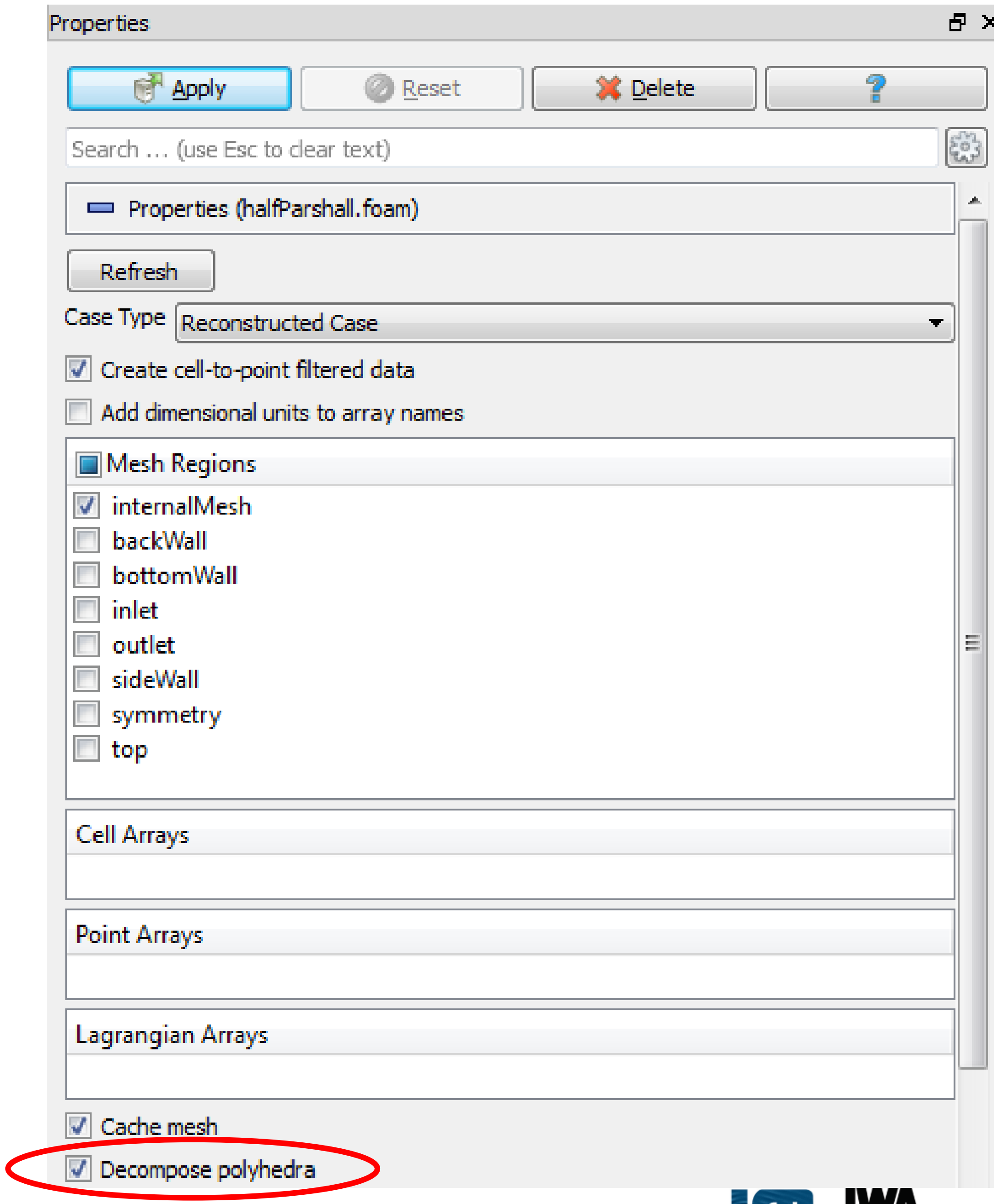
1. Make sure you load the correct time step.
   - This is because when **snappyHexMesh** is executed without the option "`-overwrite`", there will be a time snapshot for each meshing step.
2. Make sure you uncheck the option "`Decompose polyhedra`" (shown in the next slide).
3. Make sure you use the filter "`Extract Cells by Region`".
   - This is because the "`Slice`" filter will cut the cells by decomposing the cut cells into triangles.

snappyHexMesh (49/49)

Visualisation (2/2):

If the option “Decompose polyhedra” is checked (shown at the bottom of the image), the result is that polyhedral cells will be decomposed into tetrahedral cells.

Note: Also uncheck “Cache mesh”, if you want to be able to simply click on the “Refresh” button for seeing a newly generated mesh.



Case Type: Reconstructed Case

[✓] Create cell-to-point filtered data
[ ] Add dimensional units to array names

Mesh Regions
[✓] internalMesh
[ ] backWall
[ ] bottomWall
[ ] inlet
[ ] outlet
[ ] sideWall
[ ] symmetry
[ ] top

[✓] Cache mesh
[✓] Decompose polyhedra

Virtual Tracer Tests: Coupling CFD and CREng to Simulate WRRFs Unit Processes

Getting Started with OpenFOAM

nelson.marques@fsdynamics.pt; bruno.santos@fsdynamics.pt

1st September 2019

Watermatex 2019
DTU
Danmarks Tekniske Universitet
LUND UNIVERSITY
IWA the international water association
1 – 4 September 2019 | Copenhagen - Denmark
WATERMATEX 2019
10th IWA Symposium on Modelling and Integrated Assessment
Photo: Jacob Schjørring and Simone Lau, Copenhagen Media Center

FS DYNAMICS optimises your technology

Simulation

Meshing et al

Section Contents

Relevant solvers (1/3)

- OpenFOAM has a number of solvers available: openfoam.org/features/
- With OpenFOAM, users select solvers rather than models, as in commercial CFD codes.
- There are basic modelling functionalities which permeate several solvers at once. For example: solvers that allow for turbulence modelling generally allow users to select one model from the vast available range; the same is true for the specification of thermophysical properties.
- New solvers can be easily (relatively speaking) developed if necessary, since OpenFOAM is fundamentally a tool for solving partial differential equations.

Relevant solvers (2/3)

“Basic” CFD codes
- *laplacianFoam*
- *scalarTransportFoam*
- *potentialFoam*
- *etc*

Incompressible Flow
- *icoFoam*
- *nonNewtonianIcoFoam*
- *etc*

Compressible Flow
- *rhoPimpleFoam*
- *rhoSimpleFoam*
- etc

Multiphase flow
- *bubbleFoam*
- *interFoam*
- *MRFMultiphaseInterFoam*
- etc

Combustion
- *reactingFoam*
- etc

Particle-Tracking flows
- *UncoupledKinematicParcelFoam*
- *LTSReactingParcelFoam*
- etc

Heat transfer
- *buoyantBoussinesqPimpleFoam*
- *buoyantPimpleFoam*
- *etc*

Molecular dynamics
- *etc*

DNS

Electromagnetics

Stress analysis of solids

Finance

optimises your technology

Watermatex 2019

the international water association

Relevant solvers (3/3)

Solver	Base model equations	Other physical models	Turbulence	Time domain	Fluid properties
interFoam	Navier-Stokes	VOF	All incompressible flow models	Transient	Constant
buoyantBoussinesqPimpleFoam	Navier-Stokes	Energy equation, with radiation	All incompressible flow models	Transient	
driftFluxFoam	Navier-Stokes	Settling Scalar(s)	Incompressible flow models	Transient	Varying

Boundary conditions (1/21)

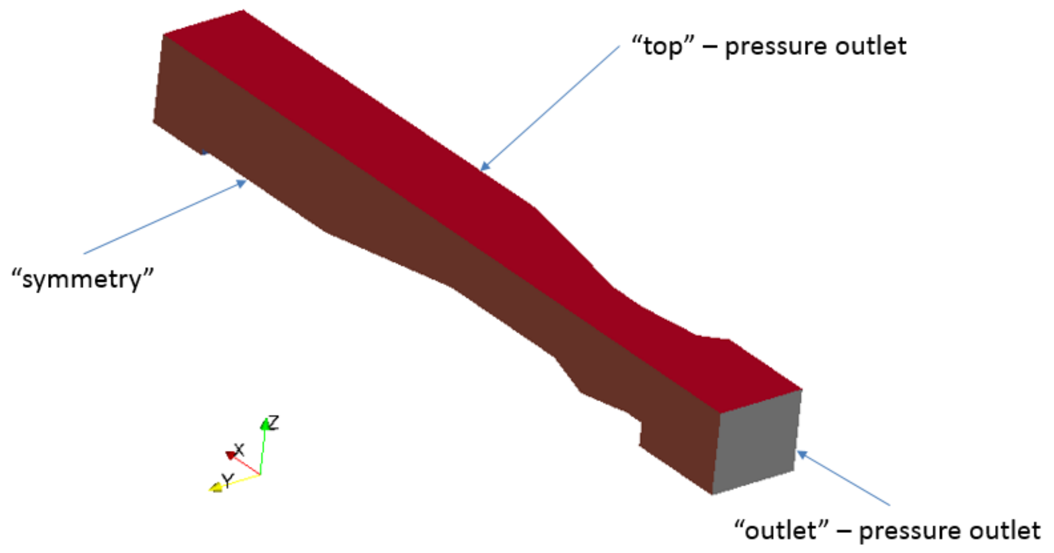
We will show:

1. How to define boundary conditions for the fields at *t= 0s*
2. How to initialize the internal fields
3. How to set-up the turbulence models and initial values

The case overview and respective boundary conditions are quickly revised in the next slide:

Boundary conditions (2/21)

- Case name: **halfParshall**
- Boundary conditions:
  - Inlet: **375 kg/s**
  - Bottom floor and side wall: **no-slip**
  - Outlet surfaces: **pressure outlet**
  - Symmetry plane surface: **symmetry**
- Fluid properties:
  - Water:
    - Density: **999 kg/m^3**
    - Dynamic Viscosity: **1.15E-3 Pa.s**
  - Air:
    - Density: **1.18 kg/m^3**
    - Dynamic Viscosity: **1.855E-5 Pa.s**



Boundary conditions (3/21)

In summary, the example case has:

- 6 field files, which are initially defined in the folder **0.orig**:
    - **U** – velocity field
    - **p_rgh** – pressure field, without the hydrostatic term
    - **alpha.water** – phase fraction field
        - 1 = 100% water
        - 0 = 100% not water (air in our case)
    - **epsilon** – turbulent dissipation rate field
    - **k** – turbulent kinetic energy field
    - **nut** – turbulent dynamic viscosity field

Boundary conditions (4/21)

- 4 major groups of boundary conditions:

  1. *Inlet* – assigned to the "`inlet`" surface
     - Velocity set to a predefined value.
     - Pressure set to zero gradient or fixed flux.
     - Phase fraction set to 1.
     - Turbulence fields `k` and `epsilon` set with appropriate values (addressed in the respective subtopic section).
     - `nut` set to `calculated`.

(continues in the next slide…)

Boundary conditions (5/21)

2. *Outlet* – assigned to the "`outlet`" and "`top`" surfaces
   - Velocity set to zero gradient and/or a condition that disables recirculation.
   - Pressure (`p_rgh`*)* set to 0 Pa.
   - Turbulence fields `k` and `epsilon` set to zero gradient and a condition that disables recirculation.
   - `nut` set to `calculated`.

(continues in the next slide…)

Boundary conditions (6/21)

3. *Wall* – assigned to the “`backWall`”, “`bottomWall`”, “`sideWall`” surfaces
    - Velocity set to no-slip, i.e. 0 m/s.
    - Pressure set to zero gradient or fixed flux.
    - Turbulence fields `k`, `epsilon` and `nut` set use wall treatments.

4. *Symmetry* – assigned to the “`symmetry`” surface
    - Boundary set to “`symmetry`” on all fields.

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Boundary conditions (7/21)

Main parameters in a field file:

- `dimensions` – the units for the field:
  - Mass - kilogram
  - Length - metre
  - Time - second
  - Temperature - Kelvin
  - Quantity - mole
  - Current - ampere
  - Luminous intensity – candela
- `internalField` – the value list for the internal field
- `boundaryField` – the list of boundary conditions

Example:

[0 1 1 0 0 0 0] = m/s

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Boundary conditions (8/21)

For example, the *U* field file roughly looks like this:

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0.0 0.0 0.0);
boundaryField
{
    backWall
    {
        type                fixedValue;
        value               uniform (0.0 0.0 0.0);
    }

    bottomWall
    {
        type                fixedValue;
        value               uniform (0.0 0.0 0.0);
    }
  …
}
```

Boundary conditions (9/21)

Figuring out what are the corresponding boundary conditions is usually done with the following strategies:

1. Looking into the tutorial cases that OpenFOAM has.
2. Checking the OpenFOAM User Guide, section “5.2 Boundaries”.
3. Looking at the complete list of boundary conditions, available in the Doxygen generated code documentation: cpp.openfoam.org/v4/
   - In the section “Using OpenFOAM” are 3 links:
     - FunctionObjects namespace Foam::functionObjects
     - Boundary Conditions

optimises your technology

Watermatex 2019

the international water association

Boundary conditions (10/21)

The next slides show the boundary conditions used in the example case.

Firstly, however, a small side note about *Regular Expressions*:

- These are search patterns that OpenFOAM supports in several dictionary files.
- For example:
  - "`(backWall|bottomWall|sideWall)`" → refers to the 3 patch names `backWall`, `bottomWall` and `sideWall`.
  - "`procBoundary.*`" → refers to all patch names that start with "`procBoundary`".

For more details: en.wikipedia.org/wiki/Regular_expression

Boundary conditions (11/21)

Inlet group (1/2):

U:

```
inlet
{
  type           flowRateInletVelocity;
  massFlowRate 375;
  rho            rho;
  rhoInlet       999.0;
}
```

alpha.water:

```
inlet
{
  type  fixedValue;
  value uniform 1.0;
}
```

p_rgh:

```
inlet
{
  type  fixedFluxPressure;
  value uniform 0.0;
}
```

nut:

```
inlet
{
  type  calculated;
  value uniform 0.0;
}
```

Boundary conditions (12/21)

Inlet group (2/2):

epsilon:

```
inlet
{
  type          turbulentMixingLengthDissipationRateInlet;
  mixingLength  0.2;
  value         uniform 1.665138;
}
```

k:

```
inlet
{
  type          turbulentIntensityKineticEnergyInlet;
  intensity     0.5;
  value         uniform 3.778352;
}
```

optimises your technology

Watermatex 2019

the international water association

Boundary conditions (13/21)

Outlet group:

U:

```
outlet
{
  type  pressureInletOutletVelocity;
  value uniform (0.0 0.0 0.0);
}
```

alpha.water:

```
outlet
{
  type      inletOutlet;
  inletValue uniform 0;
  value      uniform 0;
}
```

nut:

```
outlet
{
  type  calculated;
  value uniform 0.0;
}
```

p_rgh:

```
outlet
{
  type   fixedValue;
  value  uniform 0.0;
}
```

k, epsilon:

```
outlet
{
  type   zeroGradient;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Boundary conditions (14/21)

Wall group (1/2):

U:

```
"(backWall|bottomWall|sideWall)"
{
  type   fixedValue;
  value  uniform (0.0 0.0 0.0);
}
```

alpha.water:

```
"(backWall|bottomWall|sideWall)"
{
  type  zeroGradient;
}
```

p_rgh:

```
"(backWall|bottomWall|sideWall)"
{
  type  fixedFluxPressure;
  value uniform 0.0;
}
```

nut:

```
"(backWall|bottomWall|sideWall)"
{
  type   nutkWallFunction;
  value  uniform 0.0;
}
```

Boundary conditions (15/21)

Wall group (2/2):

epsilon:

```
"(backWall|bottomWall|sideWall)"
{
  type   epsilonWallFunction;
  value  uniform 1.665138;
}
```

k:

```
"(backWall|bottomWall|sideWall)"
{
  type   kqRWallFunction;
  value  uniform 3.778352;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Boundary conditions (16/21)

Symmetry group, for all 6 fields:

```
symmetry
{
  type symmetry;
}
```

Special group, interfaces between domains, in all 6 fields:

```
"procBoundary.*"
{
  type                processor;
  value               uniform init_value_or_vector;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Boundary conditions (17/21)

Manipulation of fields and domains (1/2):

This was not used in our example case but the idea is simple:

> What if we need to initialize a part of the internal field and/or fixed value patches with a value specific only to a group of cells or faces?

This is where `setFields` comes into play. This utility will use the settings given in the dictionary file “`system/setFieldsDict`”, for assigning values to each desired field.

Example in the next slide.

Boundary conditions (18/21)

Manipulation of fields and domains (2/2):

```
defaultFieldValues
(
  volScalarFieldValue alpha.water 0
);

regions
(
  // Set cell values
  // (does zerogradient on boundaries)
  boxToCell
  {
    box (-2.0 -2.0 -1) (11.0 1.0 0.2);
    fieldValues
    (
      volScalarFieldValue alpha.water 1
    );
  }
    // Set patch values (using ==)
    boxToFace
    {
      box (-2.0 -2.0 -1) (11.0 1.0 0.2);
      fieldValues
      (
        volScalarFieldValue alpha.water 1
      );
    }
);
```

Note: The selection box is meant to include the cell centres and/or face centres, for selecting the respective cells and faces.

FS DYNAMICS optimises your technology

Boundary conditions (19/21)

Turbulence Models (1/3):

Two categories of files have to be taken into account:

- "`constant/turbulenceProperties`" – for defining the major group of turbulence modelling to be used and the respective options settings for that model.
- In the time folders, we then have the fields associated to the turbulence model we want to use.
  - Example in our "`0.orig`" folder are: `k`, `epsilon` and `nut`

optimises your technology

Watermatex 2019

the international water association

Boundary conditions (20/21)

Turbulence Models (2/3):

Content of “`constant/turbulenceProperties`”:

```
simulationType  RAS;

RAS
{
    RASModel                kEpsilon;
    turbulence              on;
    printCoeffs             on;
}
```

Want laminar flow modelling?

```
simulationType laminar;
```

optimises your technology

Watermatex 2019

the international water association

Boundary conditions (21/21)

Turbulence Models (3/3):

The last critical detail for turbulence models is:

What initial values should we use and what values at the inlets?

This depends on your simulation, but a few guidelines exist, e.g. online:

- www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions
- support.esi-cfd.com/esi-users/turb_parameters/

The bottom line is that you will have to test which values are suited to your simulation.

Convective term discretization (1/4)

Convective term refers to the divergence operator $\nabla \cdot$

The OpenFOAM Programmer's Guide goes into more details about this in the subsection *"2.4.2 The convection term"*, in which the following expression can be found:

$$\int_V \nabla \bullet (\rho \mathbf{U} \phi)\, dV = \int_S d\mathbf{S} \bullet (\rho \mathbf{U} \phi) = \sum_f \mathbf{S}_f \bullet (\rho \mathbf{U})_f \phi_f = \sum_f F \phi_f$$

Which shows how the convection term is integrated and linearized.

We will be addressing on this topic how we can control this term.

Convective term discretization (2/4)

The settings for the divergence schemes in the file "`system/fvSchemes`", namely in the block "`divSchemes`".

From our example case:

```
divSchemes
{
    default              none;
    div(rhoPhi,U)        Gauss upwind;
    div(phi,alpha)       Gauss upwind;
    div(phirb,alpha)     Gauss upwind;
    div(phi,k)           Gauss upwind;
    div(phi,epsilon)     Gauss upwind;
    div((muEff*dev(T(grad(U))))) Gauss linear;
}
```

This is currently defined to be mostly of first order discretization, i.e. `upwind`.

Convective term discretization (3/4)

How do we know which terms we need?

We either:

1. Let the solver complain when they are not present.
2. Or we look at the source code. For example:

```
cat $FOAM_SOLVERS/incompressible/simpleFoam/UEqn.H
```

we can see this:

```
// Momentum predictor
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff(U)
  ==
    fvOptions(U)
);
```

Refers to this entry:

`div(phi,U)`

Convective term discretization (4/4)

How can we increase accuracy in our example case?

- Use an intermediate scheme between *linear* and *upwind*:

```
div(rhoPhi,U)    Gauss linearUpwind grad(U);
```

- Special limiter, great for VOF-related fields:

```
div(phi,alpha)  Gauss vanLeer;
```

- Second order scheme:

```
div(phirb,alpha) Gauss linear;
```

- Problems with running in parallel with *linearUpwind*?

```
div(phi,epsilon)  Gauss linearUpwind limitedGrad;
```

along with the gradSchemes block having this entry:

```
limitedGrad     cellLimited Gauss linear 1;
```

Diffusive term discretization (1/4)

Diffusive term refers to the Laplace operator ∇^2

The OpenFOAM Programmer's Guide goes into more details about this in the subsection *"2.4.1 The Laplacian term"*, in which the following expression can be found:

$$\int_V \nabla \bullet (\Gamma \nabla \phi)\, dV = \int_S d\mathbf{S} \bullet (\Gamma \nabla \phi) = \sum_f \Gamma_f \mathbf{S}_f \bullet (\nabla \phi)_f$$

Which shows how the diffusion term is integrated and linearized.

We will be addressing on this topic how we can control this term.

Diffusive term discretization (2/4)

The settings for the Laplacian schemes in the file “`system/fvSchemes`”, in the block “`laplacianSchemes`”.

From our example case:

```
laplacianSchemes
{
    default             Gauss linear corrected;
}
```

Meaning:

- The same setting is used for all Laplacian terms.
- Second order or above should always be used.
- `corrected` option is the surface normal gradient scheme to be used.

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Diffusive term discretization (3/4)

Other examples from OpenFOAM’s tutorials:

```
Gauss linear limited corrected 0.5;
Gauss linear limited corrected 0.333;
```

These usually depend on how orthogonal or non-orthogonal the mesh cells are.

```
Gauss linear orthogonal;    =    limited corrected 0.5
Gauss linear corrected;     =    limited corrected 1
Gauss linear uncorrected;   =    limited corrected 0
```

optimises your technology

Watermatex 2019

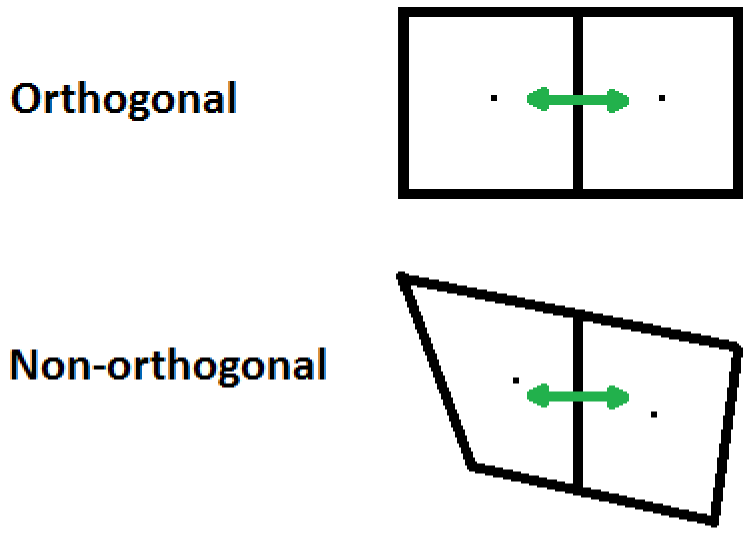
the international water association

Diffusive term discretization (4/4)

Laplacian schemes depend on the surface normal gradient discretization (block "`snGradSchemes`").

Therefore we have to take into account how the normal of a face relates to the centers of the cells that share said face.

Specifically, an orthogonal mesh has cell centers aligned with the centers and normals of the faces shared by those cells.



Linear system solvers (1/35)

The end result of the discretization process are linear systems of equations:

$$\boldsymbol{Ax} = \boldsymbol{b}$$

Where:

- the bold forms designate tensor quantities,
- uppercase letters stand for matrices,
- lowercase for vectors.

These equations will appear for every conservation law and at every outer iteration. Depending on the cases, their approximate solution can easily reach 75% of the overall CPU time.

Linear system solvers (2/35)

In order to configure the solver entry for each named block, we need to assess the type of equation that will be discretised and that will give rise the matrix form $\boldsymbol{Ax} = \boldsymbol{b}$, where:

- $\boldsymbol{A}$ is the coefficient matrix that correlates the values between the centres, i.e., our unknowns;
- $\boldsymbol{x}$ is the vector that represents the values at the cell centres for which we are solving the linear system;
- $\boldsymbol{b}$ keeps the source terms for each respective cell.

Linear system solvers (3/35)

As a result of this construct, the following types of equations will exist and the respective matrix solvers should be used:

- The equation for the pressure field, i.e., continuity equation, gives rise to a symmetric matrix, hence it should use solvers devised for this type of matrices.
- All other equations give rise to a usually non-symmetric matrix due to convection, which is why we cannot use solvers that are meant for symmetric matrix equations.

Linear system solvers (4/35)

Symmetric ***A*** matrix

$$\begin{bmatrix} i & kk & 0 & \cdots & 0 & 0 & 0 \\ kk & j & hh & \cdots & 0 & 0 & 0 \\ 0 & hh & k & \ddots & & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & & \ddots & \ddots & gg & 0 \\ 0 & 0 & 0 & \cdots & gg & r & ll \\ 0 & 0 & 0 & \cdots & 0 & ll & s \end{bmatrix}$$

Asymmetric ***A*** matrix

$$\begin{bmatrix} i & hj & qw & \cdots & 0 & 0 & 0 \\ kz & j & kq & \cdots & 0 & 0 & 0 \\ we & le & k & \ddots & & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & & \ddots & \ddots & ae & 0 \\ 0 & 0 & 0 & \cdots & ax & r & qu \\ 0 & 0 & 0 & \cdots & 0 & lw & s \end{bmatrix}$$

Linear system solvers (5/35)

The file "`system/fvSolution`" must be used to select whatever method is found appropriate to solve.

Here are a few expressions that we will be using:

- **outer iteration** – this usually refers to one step in time, if transient, or to a sweep of all transport equations which are being solved.
- **matrix solver iteration** – In the next slide you will see at the end of each line the information "`No Iterations`", which refers to the number of iterations it took to solve the respective linear system of equations.

Note: More details are available in section "4.5 Solution and algorithm control" in the OpenFOAM User Guide

Linear system solvers (6/35)

This is an example output for an **outer iteration**:

```
Time = 2

DILUPBiCG:  Solving for Ux, Initial residual = 0.33783062, Final residual = 0.022319355, No Iterations 2
DILUPBiCG:  Solving for Uy, Initial residual = 0.16839243, Final residual = 0.0037979544, No Iterations 2
DILUPBiCG:  Solving for Uz, Initial residual = 0.17283387, Final residual = 0.016074394, No Iterations 1
DILUPBiCG:  Solving for h, Initial residual = 0.97900298, Final residual = 0.020167345, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.90628728, Final residual = 0.0064221651, No Iterations 12
time step continuity errors : sum local = 0.00046531931, global = 1.4428116e-006, cumulative = -3.528428e-006
rho max/min : 1.1274839 0.72937754
DILUPBiCG:  Solving for epsilon, Initial residual = 0.99931859, Final residual = 4.869335e-005, No Iterations 1
DILUPBiCG:  Solving for k, Initial residual = 0.82482828, Final residual = 9.5040752e-006, No Iterations 1
ExecutionTime = 2.509 s  ClockTime = 2 s
```

Zooming-in on one equation:

```
DILUPBiCG:  Solving for Ux, Initial residual = 0.33783062,
Final residual = 0.022319355, No Iterations 2
```

Linear system solvers (7/35)

`system/fvSolution`:

This dictionary file was designed to handle the settings for the linear equation solvers and the algorithms to be used by a solver application, e.g. **interFoam**.

Starting with the linear equation solvers, these are configured inside this block list:

```
solvers
{
   …
}
```

It's in here that we will be configuring the matrix solvers.

optimises your technology

Watermatex 2019

the international water association

Linear system solvers (8/35)

Starting with our example case, for configuring the linear equation solvers for the fields named “alpha.water”, we are using the following settings:

```
"alpha.water.*"
{
  nAlphaCorr       2;
  nAlphaSubCycles  1;
  cAlpha           1;

  MULESCorr        yes;
  nLimiterIter     3;

  solver           smoothSolver;
  smoother         symGaussSeidel;
  tolerance        1e-8;
  relTol           0;
}
```

Specific for the linear equations related to the phase fraction equations

Linear system solvers (9/35)

Keep in mind that each solver has its own settings, where the first tier of possible options for a matrix solver refers to one of two major possible types of settings:

- **preconditioner** is needed for solvers that rely on a preconditioning strategy to speed up their iterative process.
  - For more details on what preconditioning is, see: en.wikipedia.org/wiki/Preconditioner
- **smoother** is designed to smooth-out numerical issues that usually arise from ill-formed matrices and strongly uneven intermediate solutions for the matrix equation.

More details available at: www.tfd.chalmers.se/~hani/kurser/OS_CFD_2008/TimBehrens/tibeh-report-fin.pdf

Linear system solvers (10/35)

There are 3 other parameters that are common to most of the matrix solvers. Let us look at an example output from an outer iteration:

```
DILUPBiCG:  Solving for Ux, Initial residual = 0.33783062,
Final residual = 0.022319355, No Iterations 2
```

The residual is essentially the result from this expression:

$$residual = sum(abs(\boldsymbol{b} - \boldsymbol{Ax}))$$

The reported residual values are normalized values from this equation, in order to keep values between 0.0 and 1.0 for an easier interpretation of how *good or bad* the residuals are.

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Linear system solvers (11/35)

The 3 major parameters we need to control:

- `tolerance` – this is the minimum residual value we want to achieve at the end of the iterations of the matrix solver. In other words, if the `Final Residual` falls below this value, the matrix solver stops iterating. Default value is `1e-6`.
- `relTol` – this relative tolerance refers to whether the residual for the current iteration is lesser than **relTol** times the `Initial Residual`. Default value is `0.0` (= off).
- `maxIter` – maximum number of iterations for the matrix solver to perform, regardless of the convergence status. Anti-infinite loop counter-measure. Default value is `1000`.

Linear system solvers (12/35)

One tolerance will suffice to stop the matrix solver:

- In order to only define the relative tolerance:

```
tolerance        0.0;
relTol           0.01;
```

- In order to only define the (absolute) tolerance:

```
tolerance        1e-06;
relTol           0.0;
```

- In order to allow the maximum number of iterations to be reached:

```
tolerance        0.0;
relTol           0.0;
```

Linear system solvers (13/35)

Which values should you use?

It all depends on:

- the problem you are solving;
- how accurate you want it to be, while weighing:
  - it is usually never possible to reach the exact solution for a matrix equation…
  - and even if it is, the solution might be useless if an outer iteration is still needed for balancing the results over all equations.

Therefore, this is usually something that can be adjusted after reaching good solutions for your cases.

Linear system solvers (14/35)

A few good reference values are as follows:

- For the pressure field, make sure you have a tighter control, such as:

```
tolerance         1e-06;
relTol            0.001;
maxIter           250;
```

- For all other equations, you can loosen up a bit the control, since most other equations will be affected by the pressure field in the next major iteration:

```
tolerance         1e-05;
relTol            0.01;
maxIter           100;
```

Linear system solvers (15/35)

Which matrix solvers should we use and in which situations? The answer strongly depends:

- on the simulation being performed;
- on whether the run in serial or in parallel.

Therefore, don't assume that there is a fool proof way of selecting the solvers and respective preconditioners or smoothers.

Nonetheless, it is possible to present some guidelines:

1. When in doubt, use the values in the tutorials which are the most similar to your problem.

(continues…)

Linear system solvers (16/35)

2. If you need the matrix solving steps to be as fast/efficient as possible, a good combination is to use:

   - For the pressure equations, use “GAMG”. Example:

```
p
{
      solver                 GAMG;
      smoother               GaussSeidel;
      cacheAgglomeration true;
      nCellsInCoarsestLevel 10;
      agglomerator         faceAreaPair;
      mergeLevels            1;
      tolerance              1e-06;
      relTol                 0.001;
      maxIter                250;
};
```

(continues…)

Linear system solvers (17/35)

- GAMG can be somewhere 2 and 5 times faster than using PCG+DIC.
- There is a downside to using GAMG: it is only efficient enough if you properly calibrate its parameters. For example:
  - “nCellsInCoarsestLevel” can depend on the number of cells your case has, but make sure to test the values for a few major iterations first, before using the value in a real simulation scenario.
- The PBiCG solver is usually more efficient for running in parallel for all other equations (non-pressure). Although in some cases, GAMG is faster than PBiCG.

(continues…)

Linear system solvers (18/35)

3. The PCG solver can in many cases be better than GAMG for the pressure equations, therefore, you should always double-check which one is best for your simulation.
4. In some cases, the FDIC preconditioner may prove to be more efficient than DIC and give results faster (symmetric matrices only).
5. GAMG can also be used as a preconditioner for working cooperatively with a matrix solver, but isn't very common. But again, it can be and should be tested for your own cases.

(continues…)

Linear system solvers (19/35)

6. A choice of smoother (specific matrix solvers only), may depend strongly on your case. A few examples:
   - GaussSeidel is commonly used in conjunction with GAMG, since it can offer a direct resolution for each major block.
   - DIC and DILU can also be used as smoothers, but they can prove to be more efficient if used in conjunction with GaussSeidel, namely by using the variants DICGaussSeidel and DILUGaussSeidel.

(continues…)

Linear system solvers (20/35)

7. "smoothSolver" as a matrix solver can prove to be more efficient, if a preconditioner has issues due to numerical spikes. A smoother will instead try to solve the equation directly, while *sort-of not fretting over imperfections* in the achieved solutions. Configuration example:

```
U
{
    solver              smoothSolver;
    smoother            GaussSeidel;
    tolerance           1e-8;
    relTol              0.1;
    nSweeps             1;
}
```

Linear system solvers (21/35)

Continuing with the settings we have on our example case:

```
pcorr
{
  solver               GAMG;
  tolerance            1e-5;
  relTol               0.001;
  smoother             GaussSeidel;
  nPreSweeps           0;
  nPostSweeps          2;
  cacheAgglomeration on;
  agglomerator         faceAreaPair;
  nCellsInCoarsestLevel 10;
  mergeLevels          1;
}
```

```
p_rgh
{
  $pcorr;
  tolerance          1e-07;
  relTol             0.05;
}

p_rghFinal
{
  $p_rgh;
  relTol             0;
}
```

Linear system solvers (22/35)

And for all of the other fields:

```
"(U|k|epsilon).*"
{
  solver          smoothSolver;
  smoother        symGaussSeidel;
  tolerance       1e-06;
  relTol          0;
  minIter         1;
}
```

FS DYNAMICS optimises your technology

Watermatex 2019

IWA the international water association

Linear system solvers (23/35)

Algorithm configurations

The most common algorithms implemented in OpenFOAM:

- PISO – Pressure-Implicit Split-Operator
- SIMPLE – Semi-Implicit Method for Pressure-Linked Equations
- PIMPLE – it's a PISO-SIMPLE hybrid

In our case example, we use **interFoam** with these settings:

```
PIMPLE
{
  momentumPredictor   no;
  nOuterCorrectors    1;
  nCorrectors         3;
  nNonOrthogonalCorrectors 1;
}
```

Linear system solvers (24/35)

In general, the algorithms have one or more of the following parameters:

```
nOuterCorrectors    0;
nCorrectors         0;
nNonOrthogonalCorrectors 0;
turbOnFinalIterOnly   no;
momentumPredictor yes;
transonic          no;
residualControl
{
    p                   1e-2;
    U                   1e-3;
    "(k|epsilon|omega)" 1e-3;
}
pRefCell         0;
pRefPoint        (0 0 0);
pRefValue        0;
```

Linear system solvers (25/35)

Description for each parameter:

- `nOuterCorrectors` – number of iterations that should be used for the external loop of the algorithm (not to be confused with "outer loop iteration" we've been referring to for the time step).

- `nCorrectors` – number of iterations that should be used for the internal loop of the algorithm.

- `nNonOrthogonalCorrectors` – number of iterations for attempting to correct the effect that non-orthogonal cells have on the solution of the problem. Rule of thumb:
  - 0 for a fully orthogonal mesh;
  - 20 iterations for the most non-orthogonal meshes;
  - 1-3 iterations if there are a few non-orthogonal cells.

Linear system solvers (26/35)

- `turbOnFinalIterOnly` – this flag allows us to postpone the calculation of the turbulence fields to the last iteration.
- `momentumPredictor` – Not all solvers use this parameter. Those that do support this parameter, will not solve the momentum equation (*U*) if this parameter is set to “`no`”.
- `transonic` – Only solvers that have implementations for sonic flow will support this flag.
- `residualControl` - This is a named block gives the ability to add an additional stopping criteria for the *Initial Residual* of each one of the listed equations.

Linear system solvers (27/35)

Side note

The following three parameters fall in a new topic: having a location in the domain with a fixed reference pressure value.

This is necessary whenever all of pressure boundary conditions do not have a fixed value, i.e. if they are all defined as zero gradient or a similar boundary condition.

For such a situation, we must avoid having an incomplete definition of the pressure equation, therefore we can rely on the following parameters for defining a specific location in the mesh that has a fixed and pre-defined value of pressure.

Linear system solvers (28/35)

- `pRefCell` or `pRefPoint` - For selecting a location in the mesh where the cell centre is used for pressure reference. Keep in mind that:
  - `pRefCell` refers to the cell ID on the mesh.
  - `pRefPoint` ensures us that the provided position is used for selecting the cell.
  - Note: `pRefCell` takes precedence over `pRefPoint`.
- `pRefValue` - This is the pressure value, which should be defined with the same units as the pressure fields.

Linear system solvers (29/35)

Almost complete example for SIMPLE:

```
SIMPLE
{
    nNonOrthogonalCorrectors 0;

    momentumPredictor yes;
    transonic         no;

    residualControl
    {
        p                 1e-2;
        U                 1e-3;
        "(k|epsilon|omega)" 1e-3;
    }

    //pRefCell        0;
    pRefPoint       (0 0 0);
    pRefValue       0;
}
```

Depends on the solver

optimises your technology

Watermatex 2019

the international water association

Linear system solvers (30/35)

Almost complete example for PISO:

```
PISO
{
    nCorrectors         2;
    nNonOrthogonalCorrectors 0;

    momentumPredictor yes;

    pRefCell        0;
    pRefPoint       (0 0 0);
    pRefValue       0;
}
```

Depends on the solver

FS DYNAMICS optimises your technology

Watermatex 2019

the international water association

Linear system solvers (31/35)

Almost complete example for PIMPLE:

```
PIMPLE
{
    nOuterCorrectors    1;
    nCorrectors         2;
    nNonOrthogonalCorrectors 0;
    turbOnFinalIterOnly   no;
    momentumPredictor yes;
    transonic           no;
    residualControl
    {
        p                   1e-2;
        U                   1e-3;
        "(k|epsilon|omega)" 1e-3;
    }

    //pRefCell          0;
    pRefPoint        (0 0 0);
    pRefValue        0;
}
```

$\left.\begin{array}{l}\text{turbOnFinalIterOnly}\\ \text{momentumPredictor}\\ \text{transonic}\end{array}\right\}$ Depends on the solver

Linear system solvers (32/35)

Relaxation factors

Technically, OpenFOAM uses under-relaxation factors, because the values are between 0.0 and 1.0.

These help the outer iterative convergence process, making it more robust while simultaneously increasing the likelihood that we can reach a numerical solution for our problem. A poor choice of values can delay and even prevent convergence.

This is further explained in the OpenFOAM User Guide, subsection “4.5.2 Solution under-relaxation”.

Linear system solvers (33/35)

1. If the value 0.0 is given, then the solution is kept unchanged between each outer iteration.
2. If 1.0 is given, then the under-relaxation does not take place at all.
3. As we reduce the factor from 1.0 towards 0.0, we increase the impact of the under-relaxation. Examples:
   a) 0.9 – 90% of current solution of outer iteration is preserved and 10% previous outer iteration.
   b) 0.1 – the current solution has a very small impact in the final solution after this relaxation step, making the flow results to evolve slower with every outer iteration.

Linear system solvers (34/35)

Relaxation factors: transient vs steady-state

- In most cases, the relaxation factors for transient simulations are either simply set to 1.0 or not at all.
- This is because PISO and PIMPLE algorithms perform time accurate simulations, where the time step essentially does what the relaxation factor is used for in steady-state simulations.
- Nonetheless, PIMPLE is a hybrid algorithm, therefore, it can also rely on the relaxation factors for the SIMPLE loop within the PIMPLE algorithm. Therefore, those relaxation factors are still applicable.

Linear system solvers (35/35)

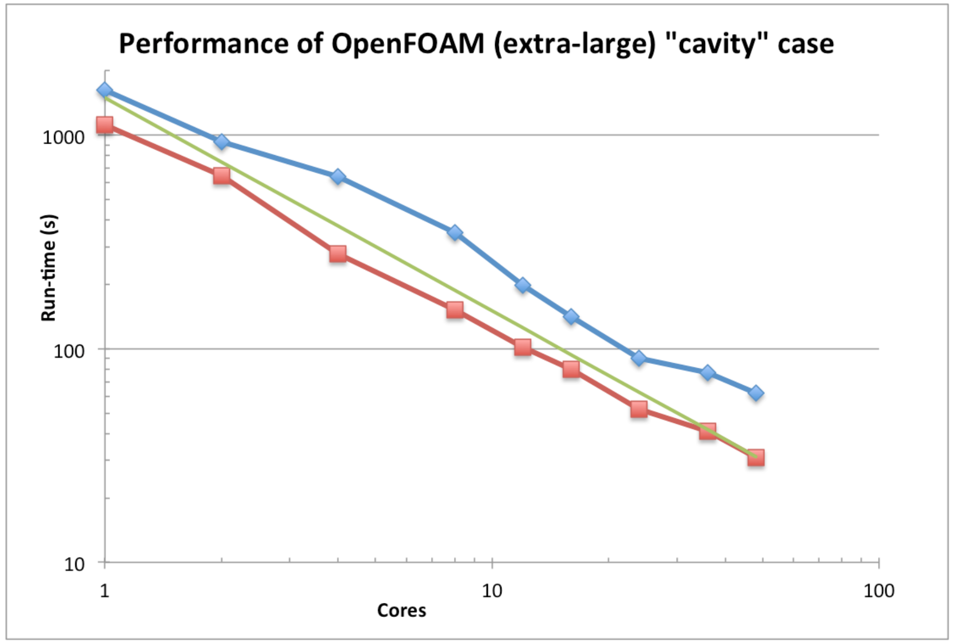
From our example case it, doesn’t need much for **interFoam** (PIMPLE):

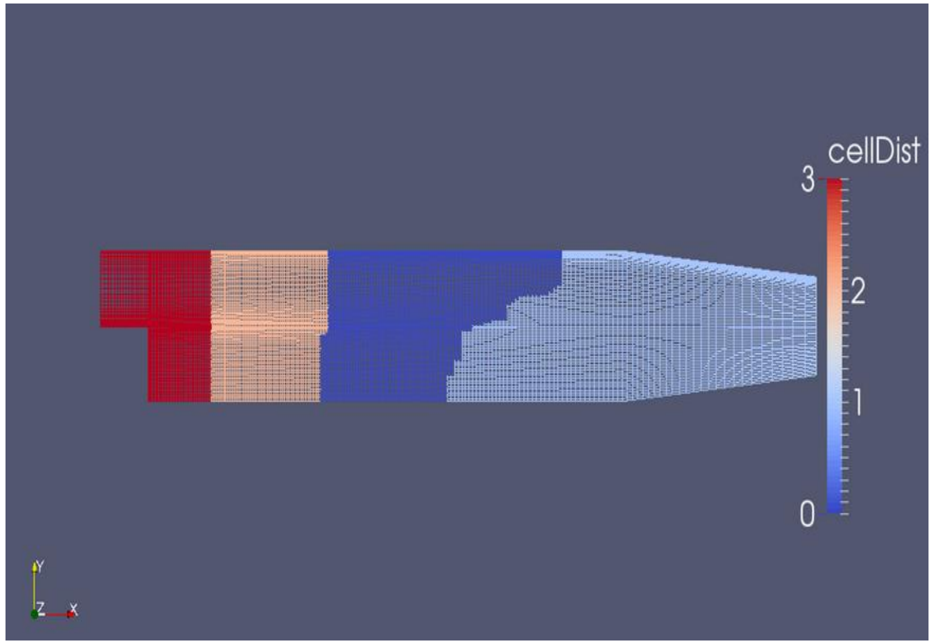
```
relaxationFactors
{
  fields
  {
  }
  equations
  {
    ".*" 1;
  }
}
```

When compared to the tutorial “`incompressible/simpleFoam/motorBike`”:

```
relaxationFactors
{
  fields
  {
    p                0.3;
  }
  equations
  {
    U                0.7;
    k                0.7;
    omega            0.7;
  }
}
```

Parallel runs (1/9)





We will address the following topics:

1. Domain decomposition
2. Domain balancing
3. Running in parallel
4. Domain reconstruction

Parallel runs (2/9)

Domain decomposition (1/2)

Application: `decomposePar`

Dictionary: `system/decomposeParDict`

Relevant parameters:

- `numberOfSubdomains` is the number for sub-domains, i.e. how many processors for running in parallel.
- `method` is for choosing the algorithm for decomposing the domain. The easiest to use is the `scotch` option.
- `scotchCoeffs` is the block relative to the method `scotch`, which doesn't even need be present, since its internal parameters are for advanced users.

Parallel runs (3/9)

Domain decomposition (2/2)

Example of “system/decomposeParDict”:

```
numberOfSubdomains 4;

method             ???;

???Coeffs
{
}
```

```
method          hierarchical;

hierarchicalCoeffs
{
    n               (1 2 1);
    delta           0.001;
    order           xyz;
}
```

OR

```
method                  scotch;

scotchCoeffs
{
    //writeGraph  true;
    //strategy "b";
}
```

Example uses:

```
    decomposePar
    decomposePar -help
    decomposePar -force
    decomposePar -cellDist
    decomposePar -noZero -fields -time 10
```

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Parallel runs (4/9)

Domain balancing

There are essentially two types of domain balancing:

1. Complete sub-domain redistribution, by using `redistributePar`, which can help balance the number of cells per processor.
    - Requires “`system/decomposeParDict`”.
    - Can run in parallel.
2. Reordering the connections between cells, by using `renumberMesh`, which will improve the configuration of the equations in matrix form, for an optimum memory access.
    - Can run in parallel.

Parallel runs (5/9)

Running in parallel (1/4)

The common denominator is that the "`-parallel`" option must be used. For example, if we run this command:

```
simpleFoam -help
```

We will see this line:

```
-parallel          run in parallel
```

Therefore, for running in parallel, the simplest command would be:

```
mpirun -np 2 simpleFoam -parallel
```

where the "`-np`" means that the number on the right is the number of processors to be used, i.e. 2.

Parallel runs (6/9)

Running in parallel (2/4)

The use of `mpirun` is not a standard on all platforms, e.g. clusters can use dedicated job schedulers and use dedicated scripts.

OpenFOAM has another two ways for running in parallel:

- `foamJob` is a script that comes in handy for running any utility and it has the ability to either run in serial or in parallel.
- `runParallel` is a function-script that is accessible only when we source the script `RunFunctions`. This is why this function is only seen inside the `Allrun` scripts that are present in OpenFOAM's tutorials.

Parallel runs (7/9)

Running in parallel (3/4) – Examples for **foamJob**:

Run in parallel as a background job:

```
foamJob -p simpleFoam
```

Run in parallel and show on-screen the output:

```
foamJob -p -s simpleFoam
```

For more details:

```
foamJob -help
```

Note: Application output is saved into the file named “`log`”.

FS DYNAMICS optimises your technology

Watermatex 2019 the international water association

Parallel runs (8/9)

Running in parallel (4/4) – Details for `runParallel`:

Will only work once this command is used (or similar) :

```
source $WM_PROJECT_DIR/bin/tools/RunFunctions
```

commonly found in the `Allrun` scripts.

Usage structure:

```
runParallel app_name number_of_cores app_arguments
```

Example:

```
runParallel snappyHexMesh 4 -overwrite
```

FS DYNAMICS optimises your technology

Watermatex 2019 the international water association

Parallel runs (9/9)

Domain reconstruction

As mentioned before, there are two types of domain reconstruction:

1. When the mesh was generated in parallel, we need to reconstruct it with `reconstructParMesh`.
2. When the mesh is the same before and after decomposing/reconstructing, then `reconstructPar` should be used for reconstructing the time snapshots.

Both can only be executed in serial mode (not in parallel).

More details with the "`-help`" option, e.g.:

```
reconstructPar -help
```

Thank you!

Any questions?

optimises your technology

Watermatex 2019

the international water association

Virtual Tracer Tests: Coupling CFD and CREng to Simulate WRRFs Unit Processes

OpenFOAM advanced topics

nelson.marques@fsdynamics.pt; bruno.santos@fsdynamics.pt

1st September 2019

Watermatex 2019
DTU
Danmarks Tekniske Universitet
LUND UNIVERSITY
IWA the international water association
1 – 4 September 2019 | Copenhagen - Denmark
WATERMATEX 2019
10th IWA Symposium on Modelling and Integrated Assessment
Photo: Jacob Schjørring and Simone Lau, Copenhagen Media Center

Section Contents

1. Power User: Coding is a must
2. The case for a GUI
3. Where next for training and services?
4. Large cases
5. Community involvement

optimises your technology

Watermatex 2019

the international water association

Power user: Coding is a must (1/2)

When we visit the OpenFOAM User Guide page, the second phase states:

> *OpenFOAM is a collection of approximately 250 applications built upon a collection of over 100 software libraries (modules). Each application performs a specific task within a CFD workflow.*

This means that it does not guaranteed that it’s able to do everything you need right out-of-the-box.

It does however provide:

- The means for creating one’s own solvers and libraries
- Given its open source nature, the people in the community may have already done what we need!

Power user: Coding is a must (2/2)

Overview of what you can do and learn:

- OpenFOAM User Guide and Programmer's Guide
- openfoamwiki.net/index.php/OpenFOAM_guide
- Learn C++:
  - Book: *Thinking in C++*
  - Tutorial: www.cplusplus.com/doc/tutorial/
- Study the source code (what matters to you)
  - openfoam.org/docs/cpp/
- Research the forums:
  - www.cfd-online.com/Forums/openfoam/

The case for a GUI (1/2)

Why doesn't OpenFOAM have a GUI?

It did, was named FoamX and it was reported to be akin to "keyhole surgery"[1] – last version was in OpenFOAM 1.4.

Depending on your workflow, a GUI can end up just getting in the way, since editing text files directly can be quicker and easier to do, as well as adding some automation to the process.

Several GUIs do already exist, which are listed here:

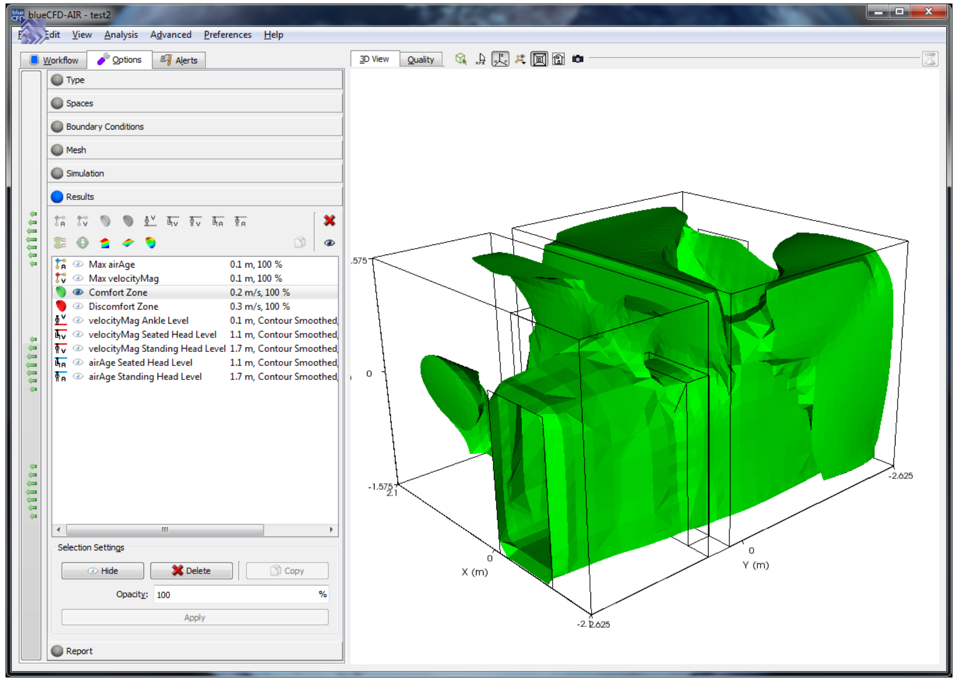
- openfoamwiki.net/index.php/GUI

[1] openfoamwiki.net/index.php/Main_UserGuideAddendum#Appendix_A:_The_FoamX_case_manager

The case for a GUI (2/2)

Pointing out a few from the list:

- CastNet by DHCAE
  - www.dhcae-tools.com/CastNet.html
- HELYX by ENGYS
  - engys.com/products/helix
- HELYX-OS (open source)
  - engys.com/products/helyx-os
- blueCFD-AIR (yes, we have one!)
  - bluecfd.com/AIR



Max airAge	0.1 m, 100 %
Max velocityMag	0.1 m, 100 %
Comfort Zone	0.2 m/s, 100 %
Discomfort Zone	0.3 m/s, 100 %
velocityMag Ankle Level	0.1 m, Contour Smoothed,
velocityMag Seated Head Level	1.1 m, Contour Smoothed,
velocityMag Standing Head Level	1.7 m, Contour Smoothed,
airAge Seated Head Level	1.1 m, Contour Smoothed,
airAge Standing Head Level	1.7 m, Contour Smoothed,

FS DYNAMICS optimises your technology

Watermatex 2019 IWA the international water association

Where next for training and services?

- OpenCFD (ESI Group) – official support providers
- CFD Direct (original developer Henry Weller, circa 1989)
- Wikki (early developer Hrvoje Jasak, circa 1993)
- Chalmers Professional Education
  - Håkan Nilsson's PhD course in CFD with OpenSource software

Complete lists:

- openfoamwiki.net/index.php/Main_Courses
- openfoamwiki.net/index.php/Template:Frontpage_Links

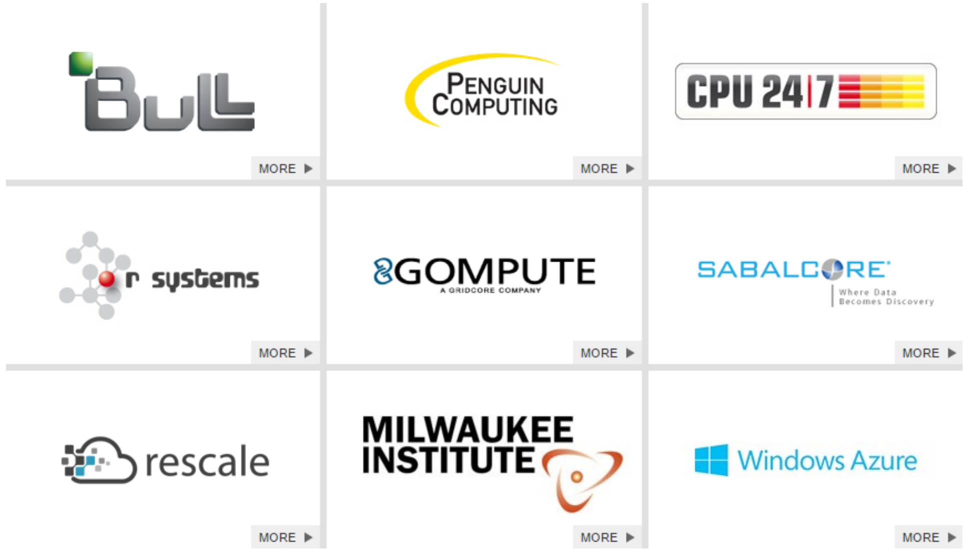
Large cases

OpenFOAM on the cloud:

- Rescale: www.rescale.com
- Gompute: www.gompute.com
- SimScale: simscale.com
- SabalCore: www.sabalcore.com
- UberCloud: www.theubercloud.com

For the DIY cloud crowd:

- Amazon EC2: aws.amazon.com/ec2 → CFDDFC Command Line Interface
- Google Cloud: cloud.google.com
- Microsoft Azure: azure.microsoft.com



Community involvement

- Forums:
  - www.cfd-online.com/Forums/openfoam/
- Wikis:
  - openfoamwiki.net
  - wiki.openfoam.com
- Contributions:
  - openfoamwiki.net/index.php/Extend-bazaar
  - openfoamwiki.net/index.php/Main_ContribOther
- Bug reports:
  - bugs.openfoam.org
  - develop.openfoam.com
  - sf.net/p/openfoam-extend/ticketsfoamextendrelease/

Thank you!

Any questions?

optimises your technology

Watermatex 2019

the international water association