Introduction to Open Source CFD for Water Resource and Recovery Facilities

Introduction

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

30th September – 1st October 2017

Objectives

• Make the case that OpenFOAM is a viable platform for CFD work in water and wastewater treatment

• Explain what OpenFOAM really is and how it is “built”

• Provide instructions on how to efficiently setup and use OpenFOAM

• Show what is available out-of-box and is relevant to WWT community
Outcomes

Attendants will:

• Have access to OpenFOAM on their computers
• Know their way around the installation
• Become familiar with OpenFOAM’s case structure (i.e., files)
• Know what steps to take to setup and run a case
• Know the most important parameters to adjust in a case setup
• Have access to four cases relevant in the WWT world

Progress – Day 1

<table>
<thead>
<tr>
<th>TIME</th>
<th>TOPIC</th>
<th>INSTRUCTOR AND AFFILIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 to 9:15</td>
<td>Welcome and Introduction to CFD for WRBF</td>
<td>Randal Samstag, Civil and Sanitary Engineer</td>
</tr>
<tr>
<td>9:15 to 10:00</td>
<td>Good Modeling Practice for CFD</td>
<td>Edward Wicklein, Carollo Engineers</td>
</tr>
<tr>
<td>10:00 to 10:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>10:30 to 12:00</td>
<td>Introduction to CFD using Open FOAM</td>
<td>Nelson Marques, blueCAPE</td>
</tr>
<tr>
<td>12:00 to 1:30</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>1:30 to 3:00</td>
<td>Getting Started with OpenFOAM: Example Case (Parshall flume) - Setup, Meshing, Pre Processing, Simulation and Post Processing</td>
<td>Nelson Marques, blueCAPE</td>
</tr>
<tr>
<td>3:00 to 3:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>3:30 to 4:00</td>
<td>CFD for flow splitting</td>
<td>Edward Wicklein, Carollo Engineers</td>
</tr>
<tr>
<td>4:00 to 5:00</td>
<td>OpenFOAM case: Flow Splitting</td>
<td>Nelson Marques, blueCAPE</td>
</tr>
</tbody>
</table>
At some point during the workshop, there is a chance you may relate a bit to this little fella!

Relax, OpenFOAM is known for being overwhelming at first. Specially for first timer users if that is your case!

On our side we have breaks and I will be around on Monday as well. Email is an option after that, so please feel free to ask to your heart’s content.

Going forward: practice makes perfect!
Next:

Introduction to CFD using OpenFOAM®

Progress

<table>
<thead>
<tr>
<th>TIME</th>
<th>TOPIC</th>
<th>INSTRUCTOR AND AFFILIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 to 9:15</td>
<td>Welcome and Introduction to CFD for WRF</td>
<td>Randal Samstag, Civil and Sanitary Engineer</td>
</tr>
<tr>
<td>9:15 to 10:00</td>
<td>Good Modeling Practice for CFD</td>
<td>Edward Wicklein, Carollo Engineers</td>
</tr>
<tr>
<td>10:00 to 10:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>10:30 to 12:00</td>
<td>Introduction to CFD using Open FOAM</td>
<td>Nelson Marques, blueCAPE</td>
</tr>
<tr>
<td>12:00 to 1:30</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>1:30 to 3:00</td>
<td>Getting Started with OpenFOAM: Example Case (Parshall flume) - Setup, Meshing, Pre Processing, Simulation and Post Processing</td>
<td>Nelson Marques, blueCAPE</td>
</tr>
<tr>
<td>3:00 to 3:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>3:30 to 4:00</td>
<td>CFD for flow splitting</td>
<td>Edward Wicklein, Carollo Engineers</td>
</tr>
<tr>
<td>4:00 to 5:00</td>
<td>OpenFOAM case: Flow Splitting</td>
<td>Nelson Marques, blueCAPE</td>
</tr>
</tbody>
</table>
OpenFOAM®
What it is and how to get it

Section Contents

1. What is OpenFOAM®?
2. Short history of OpenFOAM
3. Ecosystem around OpenFOAM
4. Operating Systems and Standards
5. What is blueCFD®-Core?
6. Installing blueCFD-Core
7. Overview of installed packages
8. Overview of installation directory
9. Getting started with the interface
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.
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 were 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/
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.

MS-DOS, Microsoft Windows, Mac OS, Mac OS X, GNU/Linux and Unix are all registered trade marks of their respective owners.

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/

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>blueCFD-Core-2016-2-win64-setup.exe</td>
<td>11/08/2016 14:14</td>
<td>Application</td>
<td>727,262 KB</td>
</tr>
</tbody>
</table>

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:

Click on the “Next” button.
Installing blueCFD-Core (3/15)

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

Notes in the next slide...
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:

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:

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:
Installing blueCFD-Core (9/15)

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

Details in the next slide...

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:

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:

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:
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.

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

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

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

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.
Overview of installed packages (3/10)

Additional Shortcuts:

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

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

**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.
- **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.
Overview of installed packages (8/10)

**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:

- **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/](http://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.

Overview of installation directory (1/3)

Double-clicking on “blueCFD-Core folder”, will go into:
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).

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).
Getting started with the interface (2/12)

Commands for file management (1/4):

```
ls  list directory contents
ls -l  same as above, but in a single column
ls -al  formatted listing with hidden files
ll  formatted listing, same as ls -l

cd dir  go to directory dir

cd  go to user home

cd ..  go back one directory

cwd  show current directory path

mkdir dir  create directory dir
```

Getting started with the interface (3/12)

Commands for file management (2/4):

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

cp filename1 filename2  copy file filename1 to file filename2

cp -r dir1 dir2  copy directory dir1 to directory dir2
```
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”

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
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

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
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
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](http://www.tldp.org)
- Linux Command website: [linuxcommand.org](http://www.linuxcommand.org)

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


---

OpenFOAM®

Bird’s-eye view
Section Contents

1. OpenFOAM Structure
2. Example Case Overview
3. Simulation Case Structure
4. Mesh Generation
5. Preprocessing
   • Model Properties
   • Boundary conditions
   • fvSolution and fvSchemes
6. Simulation
7. Post-processing

OpenFOAM Structure (1/3)

Important Environment Variables in OpenFOAM

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WM_PROJECT_DIR</td>
<td>path to the OpenFOAM installation</td>
</tr>
<tr>
<td>$WM_PROJECT_USER_DIR</td>
<td>user directory</td>
</tr>
<tr>
<td>$FOAM_TUTORIALS</td>
<td>tutorials</td>
</tr>
<tr>
<td>$FOAM_SRC</td>
<td>source code directory of OpenFOAM libraries</td>
</tr>
<tr>
<td>$FOAM_APP</td>
<td>source code directory of OpenFOAM applications</td>
</tr>
<tr>
<td>$FOAM_APPBIN</td>
<td>directory with the compiled OpenFOAM applications</td>
</tr>
<tr>
<td>$FOAM_USER_APPBIN</td>
<td>directory with the OpenFOAM applications created by the user</td>
</tr>
<tr>
<td>$FOAM_LIBBIN</td>
<td>directory with the compiled OpenFOAM libraries</td>
</tr>
<tr>
<td>$FOAM_USER_LIBBIN</td>
<td>directory with the OpenFOAM libraries created by the user</td>
</tr>
<tr>
<td>$FOAM_RUN</td>
<td>directory where the user can put his/her cases</td>
</tr>
<tr>
<td>echo variable</td>
<td>will show you the contents of environment variable, example:</td>
</tr>
<tr>
<td></td>
<td>echo $WM_PROJECT_DIR</td>
</tr>
</tbody>
</table>
OpenFOAM Structure (2/3)

Important Shell-Aliases in OpenFOAM

<table>
<thead>
<tr>
<th>foam</th>
<th>cd $WM_PROJECT_DIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>app</td>
<td>cd $FOAM_APP</td>
</tr>
<tr>
<td>sol</td>
<td>cd $FOAM_SOLVERS</td>
</tr>
<tr>
<td>tut</td>
<td>cd $FOAM_TUTORIALS</td>
</tr>
<tr>
<td>util</td>
<td>cd $FOAM_UTILITIES</td>
</tr>
<tr>
<td>src</td>
<td>cd $FOAM_SOLVERS</td>
</tr>
<tr>
<td>lib</td>
<td>cd $FOAM_LIBBIN</td>
</tr>
<tr>
<td>run</td>
<td>cd $FOAM_RUN</td>
</tr>
<tr>
<td>src</td>
<td>cd $FOAM_SRC</td>
</tr>
<tr>
<td>wmSet</td>
<td>. $WM_PROJECT_DIR/etc/bashrc</td>
</tr>
<tr>
<td>wmuUnset</td>
<td>. $WM_PROJECT_DIR/etc/config/unset.sh</td>
</tr>
</tbody>
</table>

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/m³**
    - Dynamic Viscosity: **1.15E-3 Pa.s**
  - Air:
    - Density: **1.18 kg/m³**
    - 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):

  - "sideWall" – no-slip condition
  - "bottomWall" – no-slip condition
  - "inlet" – mass flow inlet
  - "backWall" – no-slip condition
  - "outlet" – pressure outlet

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.
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

Requirement before running the solver:

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

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)

Meshing steps overview – Contents \textit{nano\_it} of the \textit{Allrun.pre} script (1/2):

\begin{verbatim}
runApplication surfaceTransformPoints -translate '(-4.25 0.687 -0.55)' \ constant/triSurface/halfParshall.org.stl \ constant/triSurface/halfParshall.stl

runApplication blockMesh
runApplication extrudeMesh
runApplication surfaceFeatureExtract

echo "decompositionMethod scotch;" > system/decomposeParDict.method
runApplication -s 1 decomposePar
echo "decompositionMethod ptscotch;" > system/decomposeParDict.method
\end{verbatim}
Mesh Generation (3/19)

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

```plaintext
runParallel snappyHexMesh -overwrite  
runApplication reconstructParMesh -constant  
runApplication createPatch -overwrite  
runApplication transformPoints -translate '(4.25 -0.687 0.55)'  
runApplication checkMesh -constant  
runApplication changeDictionary -enableFunctionEntries
```

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”.
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) – 1\textsuperscript{st} step of \texttt{snappyHexMesh}:

This is the result of the \textit{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) – 2\textsuperscript{nd} step:

This is the result of the \textit{snap} step, where it:
1. Cuts cells that overlap the geometry.
2. Morphs (snaps) the mesh onto the surface.
Seeing the mesh (5/5) – refinement detail:

This is why we needed more refinement near the outlet.

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.
Mesh Generation (15/19)

Highlights of `blockMeshDict`:

```verbatim
classToMeters 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):

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

gometry
{
  "halfParshall.stl"
  
  type triSurfaceMesh;
  regions
  {
    backWall
    {
      name backWall;
    }
    [...]
  }
  refinementBox
  {
    type searchableBox;
    min (-4.35 -1 -0.30);
    max (-4.20 1 -0.15);
  }
};
```

97

98
Mesh Generation (17/19)

Highlights of `snappyHexMeshDict` (2/4):

```plaintext
// Settings for the castellatedMesh generation.

castellatedMeshControls
{
    [...] regions
    { backWall
        { level (0 0); }
    }
    [...] top
        { level (0 0); }
    }
}

refinementSurfaces
{
    "halfParshall.stl"
    { level (0 0); }
    [...] regions
    { backWall
        { level (0 0); }
    }
    [...] top
        { level (0 0); }
    }
}

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

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

//end of castellatedMeshControls
```

Mesh Generation (18/19)

Highlights of `snappyHexMeshDict` (3/4):

```plaintext
[...]

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

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

//end of castellatedMeshControls
```
Mesh Generation (19/19)

Highlights of `snappyHexMeshDict` (4/4):

```c
// Settings for the snapping.
snapControls
{
    nSmoothPatch 3;
    tolerance 1.0;
    nSolveIter 30;
    nRelaxIter 5;
}
```

Preprocessing (1/9)

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

```c
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;
```
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.

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:

```plaintext
dimensions [0 1 -1 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:

```plaintext
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;
}
```

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;
}
```
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;
}

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(phirh, alpha) Gauss upwind;
    div(phi,k) Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div((muEff*dev(T(grad(U)))) Gauss linear;
}

Simulation (1/4)

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

```bash
./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.
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.

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.

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:

![Image of legend with "alpha.water" label]  

The legend can be moved with the mouse.

Next:

Lunch!!!