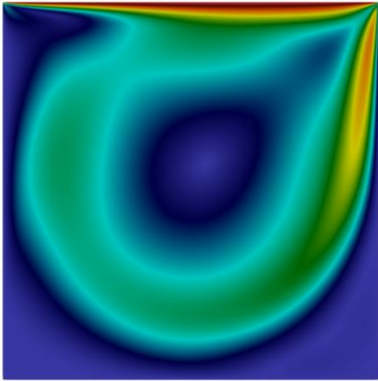




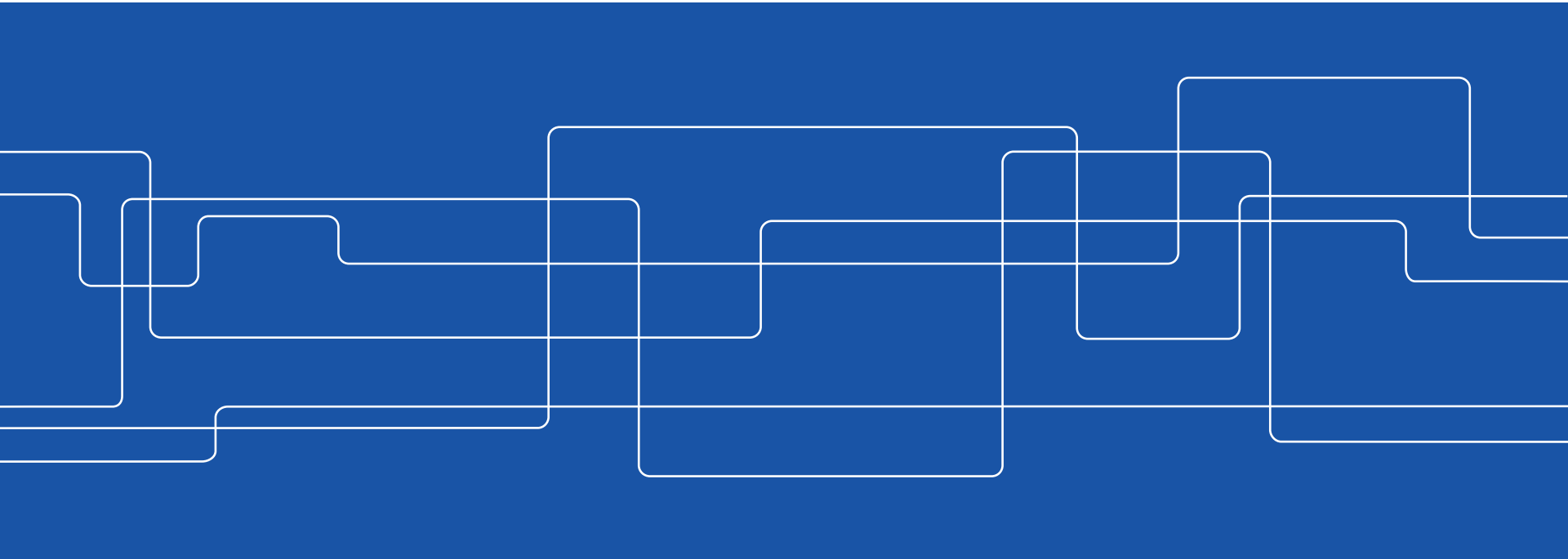
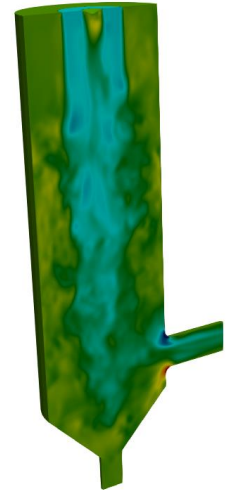
2021, Stockholm, Sweden

KTH ROYAL
INSTITUTE
OF TECHNOLOGY



Starting with *OpenFOAM*

Marco Atzori and Fermin Mallor





General information

- **Main reference:**

A tensorial approach to computational continuum mechanics using object-oriented techniques, Weller *et al.* (1998), Comput. Physics, **12**, 620

OpenFOAM uses the Finite-Volume approach.

- **Different distributions:**

1. OpenFOAM Foundation: <https://openfoam.org/>
2. ESI Group: <https://openfoam.com/>

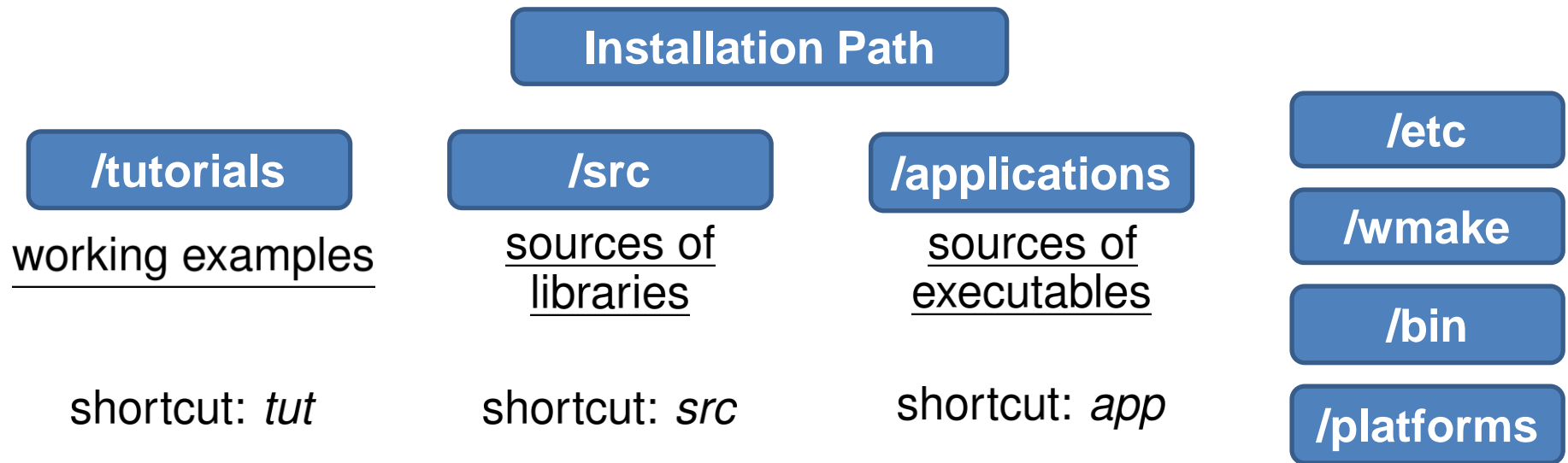


How to start:

Read the documentation!

Git repository with instructions: <https://github.com/KTH-Nek5000/tutorialOF>

Mac and Windows → use a virtual machine with Ubuntu 20





Run the first case: **Cavity**

1. Copy a tutorial from the *tutorial* folder, for instance:

/tutorials/incompressible/icoFoam/cavity/cavity

2. Create the mesh:

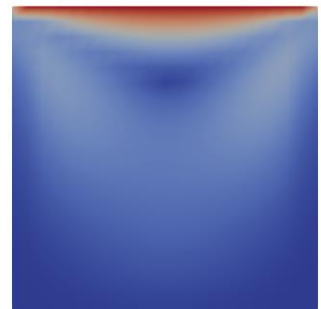
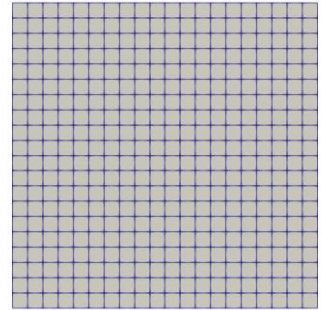
blockMesh

3. Run the solver:

icoFoam

4. Visualize with *paraView*:

paraFoam



Figures: (top) grid and (bottom) velocity magnitude.



Structure of the case (for *cavity*)

- Folder *system*:
 - **blockMeshDict**: utility to create structured grid.
 - **controlDict**: Δt , run time, outpost...
 - **fvSolution**: numerical solvers, algorithm settings...
 - **fvSchemes**: discretization of the derivatives...
- Folder *constant*:
 - **transportProperties**: physical constants, *i.e.*: ν
 - Folder *polyMesh*: the grid (*points*, *boundary*, *faces*, ...)
- Folder *0*:
 - **p**: boundary and initial conditions for p .
 - **U**: boundary and initial conditions for U .



Create structural grid: *BlockMesh*

- Set the scale.
- Define *vertices* (**the order matters!**).
- Define *blocks*, and set the discretization (either uniform or not).
- Define *boundary*

The boundaries:

- have a *name* (which is arbitrary).
- have a *type* (**the type matters!**)
- are defined as list of quadrilateral faces.



The simulation: *controlDict*

- *application*: used for builtin functions.
- *startFrom* & *startTime*: starting time (e.g.: first time or latest saved time).
- *stopAt*: e.g.: **endTime**, **nextWrite**, **writeNow**, ...
- *endTime* & *deltaT*: last time and Δt between each time step.
- *writeControl*: it is the *writeInterval* “unit”, e.g.: **timeStep** (number of time steps), **runTime** (physical time), **clockTime** (elapsed time from start), ...
- *writeInterval*: “writeControl” times between fields outpost.
- *purgeWrite*: number of outpost fields not overwritten.
- *writeFormat*, *writePrecision*, *writeCompression*, *timeFormat* & *timePrecision*: outpost formats.
- *runTimeModifiable*: if *true*, the dictionaries are read every iteration.



The simulation: *fvSchemes*

- Time derivative: (explicit) Euler.
- Pressure gradient: Gauss linear (second order).
- Divergence of ϕ, U : Gauss linear (second order).
- Laplacian: Gauss linear orthogonal (second order)
- Interpolation: linear
- (?)



The simulation: *fvSolution*

- Solver for the pressure, *e.g.* in *cavity*:
 - Preconditioned Conjugate Gradient (PCG), with simplified Diagonal-based Incomplete Cholesky (DIC) preconditioner;
 - absolute and relative tolerance.
- Solver for the velocity, *e.g.* in *cavity*:
 - symmetric Gauss Seidel smoother;
 - absolute and relative tolerance.
- Settings for the algorithm, *e.g.* *PISO* in *cavity*:
 - number of corrector steps;
 - number of non-orthogonal corrector steps;
 - reference pressure.



The algorithm: *icoFoam* (*PISO*)

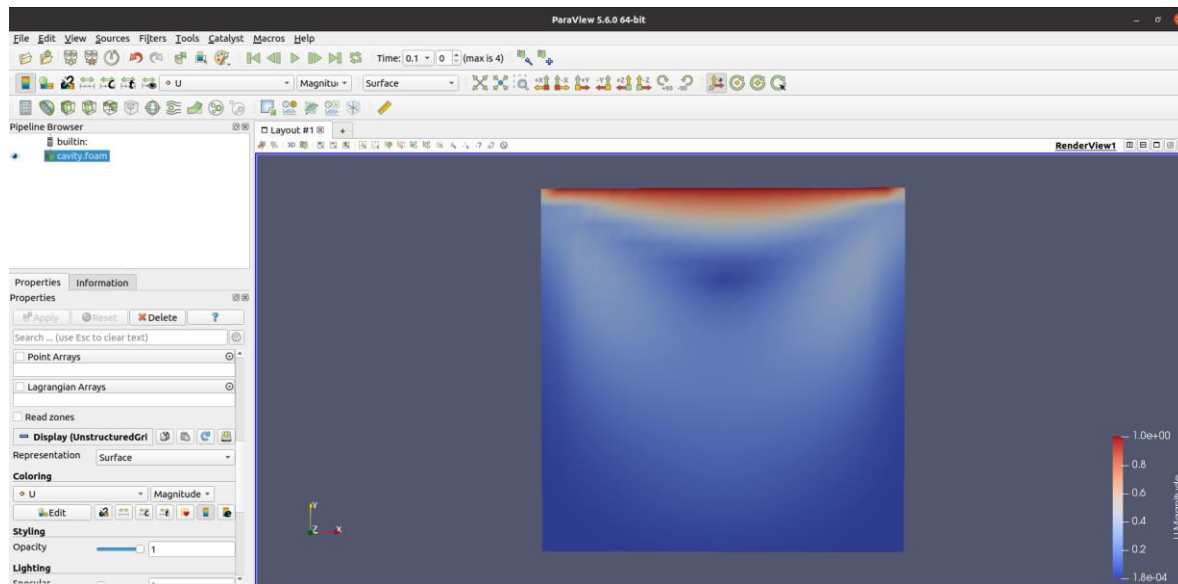
PISO (Pressure-Implicit with Splitting of Operators):

- Extension of SIMPLE algorithm
- Used mainly for unsteady flow (it is now also adapted for steady-state problems)
- 1 predictor step and 2 (or more) corrector steps to satisfy mass conservation (continuity)

(1): R.I. Issa. “Solution of the implicit discretized fluid flow equations by operator-splitting”. *J. Comp. Physics*, 62(1):40-65, 1986.

Visualization of results: *Paraview*

- Use *paraFoam* –*builtin*
- Visualization (contours, slices, line plots...) of all fields stored (U, flux, p...) at the output time-steps (or iterations)

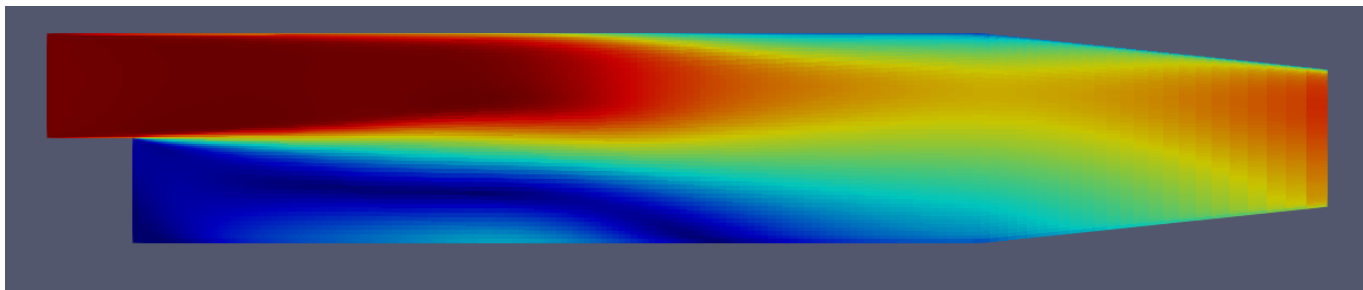




Second case: Pitz & Daily (1983)

- Backward-facing step
- RANS equations (incompressible, steady-state)
- SIMPLE algorithm (predictor + corrector steps)
- Turbulence model for closure

constant/turbulenceProperties





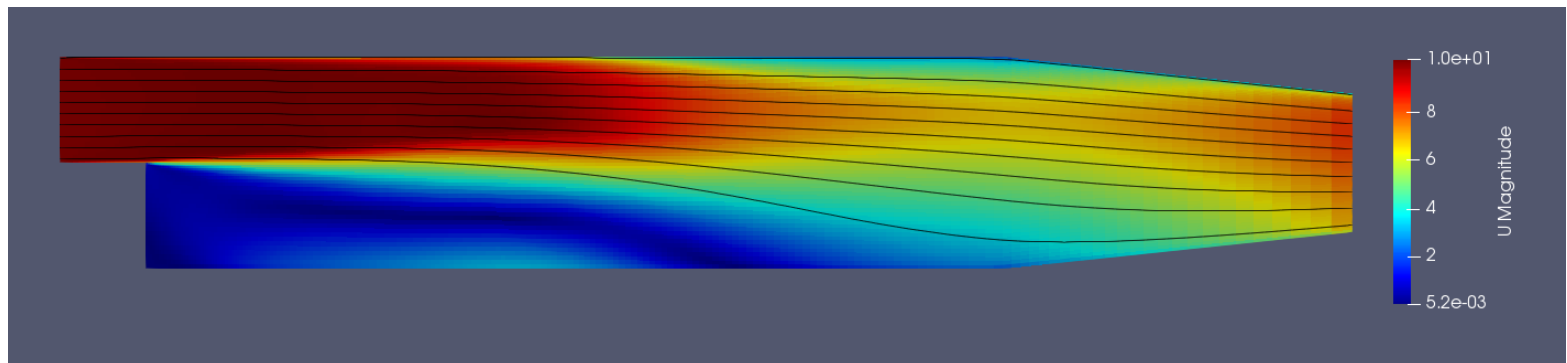
Visualization of results: *Paraview*

- *Create VTK fields using post-processing functions:*

simpleFoam -postProcess -func wallShearStress

simpleFoam -postProcess -func streamlines

- *paraFoam -builtin*





More information

- *Check the User guide (really useful!)*

<https://www.openfoam.com/documentation/user-guide/>

The screenshot shows the OpenFOAM website interface. At the top, the header includes the OpenFOAM logo, the tagline 'The open source CFD toolbox', and the esi logo. Below the header is a navigation bar with links: Home, Products, Services, Download, Code, Documentation, Community, Governance, and News. A secondary navigation bar contains links: About us, Contact, Jobs, and Legal. On the right side of the secondary navigation bar is a 'FOLLOW US ON twitter' button. The main content area is divided into two columns. The left column is titled 'User Guide' and contains a list of links: Contents, 1 Introduction, ± 2 OpenFOAM cases, ± 3 Running applications, ± 4 Mesh generation and conversion, ± 5 Models and physical properties, ± 6 Solving, ± 7 Post-processing, ± A Reference, and Index. The right column is titled 'Contents' and shows a hierarchical list of topics: 1 Introduction, 2 OpenFOAM cases (with sub-items 2.1 File structure of OpenFOAM cases, 2.2 Basic input/output file format, 2.2.1 General syntax rules, 2.2.2 Dictionaries, 2.2.3 The data file header, 2.2.4 Lists, 2.2.5 Scalars, vectors and tensors, 2.2.6 Dimensional units, 2.2.7 Dimensioned types, 2.2.8 Fields, and 2.2.9 Directives and macro substitutions), and 2.1 File structure of OpenFOAM cases.