Introduction to OpenFOAM:
OpenFOAM's history, main features and case setup

Dr. Evangelos (Vaggelis) Papoutsis-Kiachagias
Senior Researcher NTUA

School of Mechanical Engineering, NTUA,
Parallel CFD & Optimization Unit
email: vpapout@mail.ntua.gr

Main objectives of the course

- Familiarization with CFD and OpenFOAM
  - Short history
  - Capabilities and characteristics
  - Case setup

- An introduction to managing jobs in an HPC environment
  - Simulated HPC environment using the SLURM job submission system
  - Demonstration of a CFD job submission in the ARIS HPC system of GRNET

What is OpenFOAM?

- **Mainly a toolbox for solving Partial Differential Equations (PDEs)**
- **Started out in the Imperial College of London, during the 90s**
- **Started being distributed as open-source software during 2004**
- **Widely used by a variety of industries (automotive, turbomachinery, chemicals, paper, energy, etc)**
- **A wide variety of applications, mainly focused on Computational Fluid Dynamics (CFD), but not exclusively (some support for structural mechanics, finance, etc)**

What is OpenFOAM?

- Mainly a toolbox for solving Partial Differential Equations (PDEs)
- Started out in the Imperial College of London, during the 90s
- Started being distributed as open-source software during 2004
- Widely used by a variety of industries (automotive, turbomachinery, chemicals, paper, energy, etc)
- A wide variety of applications, mainly focused on Computational Fluid Dynamics (CFD), but not exclusively (some support for structural mechanics, finance, etc)

What is required by the user

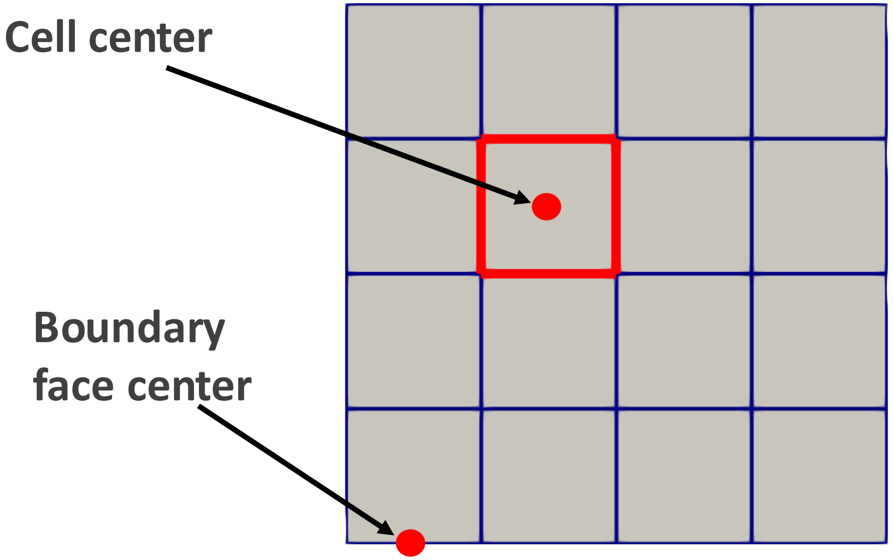
- As an open-source s/w, its “natural” environment lays in Linux
- Windows support through the Windows Subsystem for Linux (WSL)
  - We will be using OpenFOAM v2312
    https://www.openfoam.com/news/main-news/openfoam-v2312

- No Graphical User Interface (GUI)!
  - Setup and navigation of cases using the Linux Command Line Interface (CLI)
    - Some basics skills are needed (navigation through the file system, creating/copying/deleting files and folders, editing files through a text editor)
    - Some frequent CLI commands can be found here
    - Some useful CLI features of OpenFOAM can be found here

- Steep learning curve
  - A number of files need to be setup before running a case
  - A lot of freedom for the user
    - ✓ A truly versatile s/w with very few constraints
    - ✗ Freedom to setup a case with little physical meaning
  - It requires from the user to understand the problem they are trying to solve

Basic OpenFOAM characteristics

- **Based on the finite volumes method (flow equations *integrated* over control/finite volumes)**
- **Written in C++, taking full advantage of the object oriented aspects of the language (~3M lines of code)**
- **Based on a cell centered discretization of the flow equations**



Flow quantities are computed/stored at the cell centers and boundary conditions are imposed on the boundary face centers

EURO Greece

Flow equations for incompressible fluids

A plethora of solvers exists (incompressible, compressible, single- and multi-phase, mixtures, etc).
Today, we are going to focus on *incompressible* fluids (constant fluid density)

Navier-Stokes equations

$$R^p = \frac{\partial v_j}{\partial x_j} = 0$$

Flow velocity

Continuity equation: Conservation of mass

$$R_i^v = \underbrace{\frac{\partial (v_j v_i)}{\partial x_j}}_{m\alpha} \overbrace{- \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] + \frac{\partial p}{\partial x_i}}^{\sum F} = 0$$

Bulk viscosity

Pressure

Momentum equations: Newton's 2nd law of motion

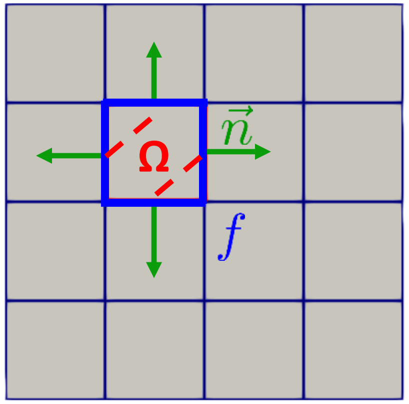
The Finite Volume Method in brief

$$R^p = \frac{\partial v_j}{\partial x_j} = 0$$

Continuity equation: Conservation of mass

$$R_i^v = \frac{\partial (v_j v_i)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] + \frac{\partial p}{\partial x_i} = 0$$

Momentum equations: Newton's 2nd law of motion



FVM = integration of the flow equations over a control (finite) volume (Ω)

Gauss divergence theorem

Discretization

$$\int_\Omega \frac{\partial v_j}{\partial x_j} d\Omega = \int_S v_j n_j dS = \sum_f \overbrace{v_j^f n_j^f dS^f}^{\text{Volume flow rate (m3/s)}}$$

Interpolating to the boundaries of the finite volume is a crucial part of FVM

EURO Greece

Segregated vs coupled solution of the flow equations

OpenFOAM solves its equations in a segregated manner

Coupled solution

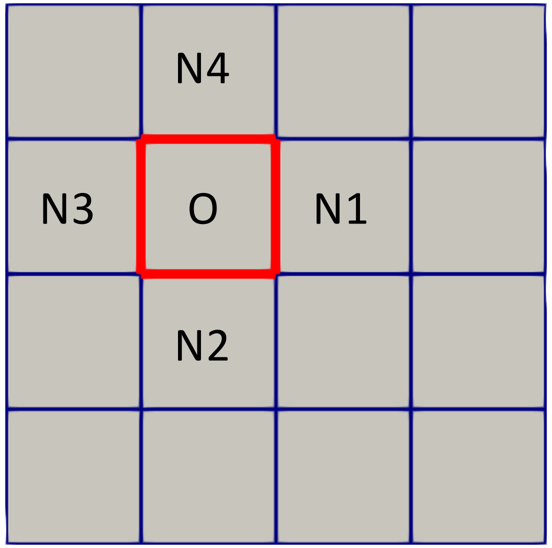
$$\begin{bmatrix} \mathbf{A}_{\mathbf{U_{xx}}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{U_{xy}}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{U_{xp}}}(\tilde{\mathbf{U}}) \\ \mathbf{A}_{\mathbf{U_{yx}}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{U_{yy}}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{U_{yp}}}(\tilde{\mathbf{U}}) \\ \mathbf{A}_{\mathbf{px}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{py}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{pp}}(\tilde{\mathbf{U}}) \end{bmatrix} \begin{bmatrix} U_x \\ U_y \\ p \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \\ b_p \end{bmatrix} \qquad \vec{U} = \begin{bmatrix} U_x \\ U_y \\ p \end{bmatrix}$$

Segregated solution

$$\begin{bmatrix} \mathbf{A}_{\mathbf{U_{xx}}}(\tilde{\mathbf{U}}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathbf{U_{yy}}}(\tilde{\mathbf{U}}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{\mathbf{pp}}(\tilde{\mathbf{U}}) \end{bmatrix} \begin{bmatrix} U_x \\ U_y \\ p \end{bmatrix} = \begin{bmatrix} b_x - \mathbf{A}_{\mathbf{U_{xy}}}(\tilde{\mathbf{U}})U_y - \mathbf{A}_{\mathbf{U_{xp}}}(\tilde{\mathbf{U}})p \\ b_y - \mathbf{A}_{\mathbf{U_{yx}}}(\tilde{\mathbf{U}})U_x - \mathbf{A}_{\mathbf{U_{yp}}}(\tilde{\mathbf{U}})p \\ b_p - \mathbf{A}_{\mathbf{px}}(\tilde{\mathbf{U}})U_x - \mathbf{A}_{\mathbf{py}}(\tilde{\mathbf{U}})U_y \end{bmatrix}$$

Implicit discretization schemes

- **Implicit discretization schemes for spatial and temporal derivatives**



Discretization of the x component of the momentum equations on cell O, with neighbors N

$$R_{U_{x,O}} = a_O U_{x,O} + \sum a_N U_{x,N} - B_x(U_x, U_y, p) = 0$$

$$\begin{bmatrix} \mathbf{A}_{\mathbf{U_{xx},O}}(\tilde{\mathbf{U}}) & \mathbf{A}_{\mathbf{U_{xx},N}}(\tilde{\mathbf{U}}) \end{bmatrix} \begin{bmatrix} U_{x,O} \\ U_{x,N} \end{bmatrix} = \begin{bmatrix} B_{x,O} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{A}_{\mathbf{U_{xx}}}(\tilde{\mathbf{U}}) & 0 & 0 \\ 0 & \mathbf{A}_{\mathbf{U_{yy}}}(\tilde{\mathbf{U}}) & 0 \\ 0 & 0 & \mathbf{A}_{\mathbf{pp}}(\tilde{\mathbf{U}}) \end{bmatrix} \begin{bmatrix} U_x \\ U_y \\ p \end{bmatrix} = \begin{bmatrix} b_x - \mathbf{A}_{\mathbf{U_{xy}}}(\tilde{\mathbf{U}})U_y - \mathbf{A}_{\mathbf{U_{xp}}}(\tilde{\mathbf{U}})p \\ b_y - \mathbf{A}_{\mathbf{U_{yx}}}(\tilde{\mathbf{U}})U_x - \mathbf{A}_{\mathbf{U_{yp}}}(\tilde{\mathbf{U}})p \\ b_p - \mathbf{A}_{\mathbf{px}}(\tilde{\mathbf{U}})U_x - \mathbf{A}_{\mathbf{py}}(\tilde{\mathbf{U}})U_y \end{bmatrix}$$

Boundary conditions

Walls:

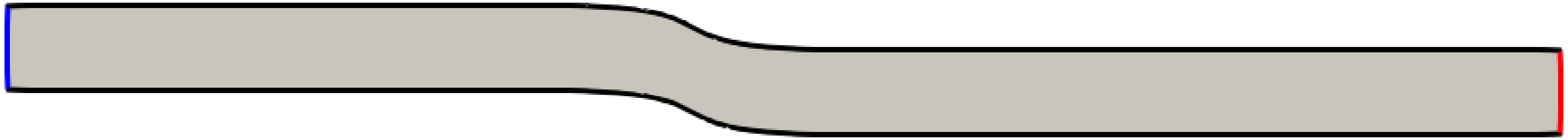
$$v_i = 0 \qquad \text{no slip condition}$$

$$\frac{\partial p}{\partial x_j} n_j = 0$$

Inlet:

$$v_i = ct$$

$$\frac{\partial p}{\partial x_j} n_j = 0$$



Outlet:

$$\frac{\partial v_i}{\partial x_j} n_j = 0$$

$$p = 0$$

- **Why impose a zero outlet pressure?**

$$R_i^v = \frac{\partial (v_j v_i)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] + \frac{\partial p}{\partial x_i} = 0$$

The pressure appears only as a gradient in the flow equations …

- **The pressure field is computed wrt a reference pressure, usually imposed at the outlet**
- **The pressure field computed in incompressible flow simulations is actually the pressure divided by the constant fluid density**

Typical structure of an OpenFOAM case (1)

- Three main folders `0, constant, system`

`0`

- Setup of the boundary conditions and the flow initialization. Should Include one file per flow field to be computed

`constant`

- `polyMesh (folder)`
  - The mesh folder. Meshes can be built using OpenFOAM's mesh generators (blockMesh, snappyHexMesh) or imported through preprocessing applications supporting the most popular mesh types
  - The `boundary` file includes the names and types of the patches of the mesh. Useful for setting up boundary conditions in the `0` folder
- `transportProperties`
  - Definition of basic fluid characteristics (e.g. viscosity)
- `turbulenceProprties`
  - Definition of the turbulence model

Typical structure of an OpenFOAM case (2)

`system:` includes dictionaries controlling our simulation

- `controlDict`
  - A number of entries controlling the number of iterations to be executed, time-step (for unsteady runs), write interval, etc
  - Can also define functions, to be executed after each iteration or at a post-processing level
- `fvSolution`
  - Definition of the linear solvers, convergence criteria, relaxation factors
- `fvSchemes`
  - Definition of the schemes used to discretize terms in the flow equations
- `decomposeParDict` (optional, pre-processing)
  - Used to decompose the mesh into sub-domains, as a prerequisite for running in parallel
- `fvOptions` (optional)
  - Defines additional source terms for the flow/adjoint equations.

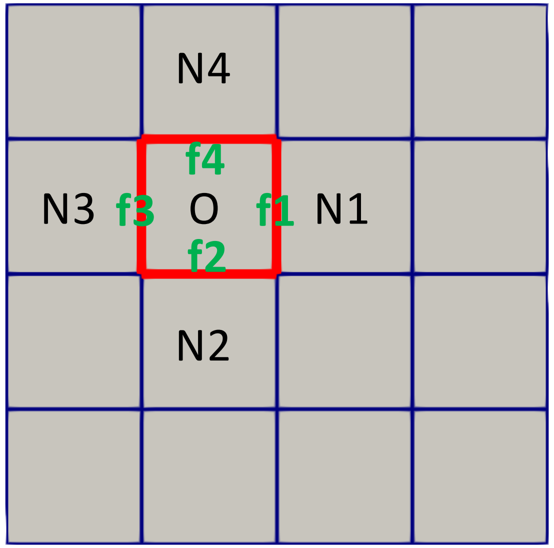
More details about the structure of an OpenFOAM case can be found here

Our first OpenFOAM case

```
>> cd 2024_06_OF_training_EuroCC_Greece/01-secondOrder
```

- Go through the content of `constant`, `0` and `system`
- Use the `slurm.sh` and `Allclean` scripts
- Post-process the results using Paraview

Discretization of the grad operator



$$R_i^v = \frac{\partial (v_j v_i)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\nu+\nu_t)\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\right] + \boxed{\frac{\partial p}{\partial x_i}} = 0$$

Computing the grad operator in cell O (e.g. grad(p)):

$$\int_\Omega \frac{\partial p}{\partial x_i} d\Omega = \int_S p n_i dS = \sum_f p^f n_i^f \Delta S^f$$

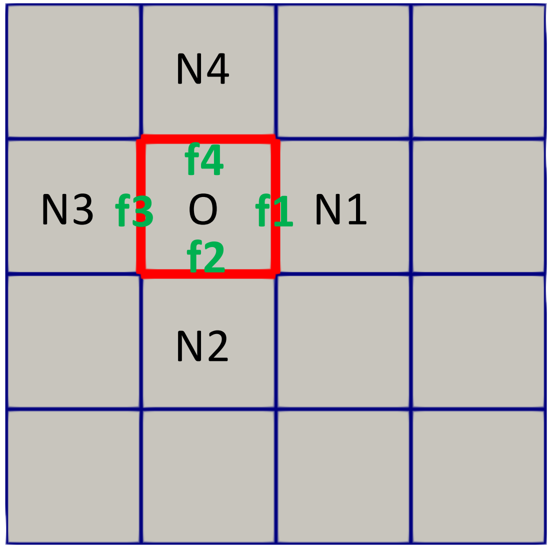
Gauss Divergence Theorem **Discretization**

$$p^{f_{O,N_1}} = w^f p^O + (1-w^f) p^{N_1}$$

Interpolation

EURO Greece

Discretization of convection terms: 1rst/2nd order schemes



$$R_i^v = \boxed{\frac{\partial (v_j v_i)}{\partial x_j}} - \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] + \frac{\partial p}{\partial x_i} = 0$$

Physical meaning of the convection term?

Interpolation??

$$\int_\Omega \frac{\partial (v_j v_i)}{\partial x_j} d\Omega = \int_S v_i (v_j n_j) dS = \sum_f v_i^f (v_j n_j)^f \Delta S^f$$

??

linear

2nd order. In OpenFOAM: linearUpwind

$$v_i^f = v_i^O$$

1rst order. In OpenFOAM: upwind

linear

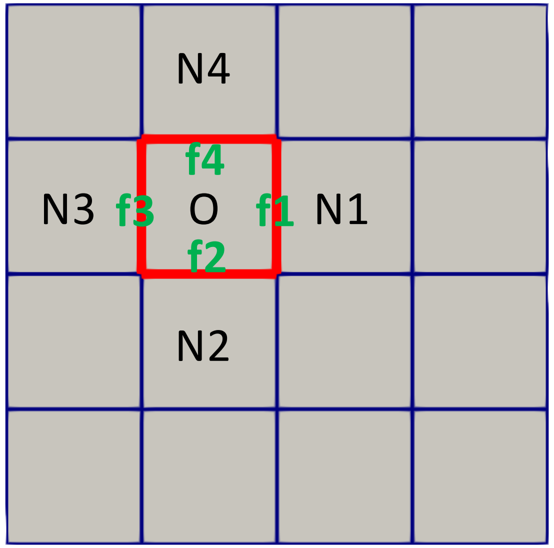
How to compute grad(U)?

$$\frac{\partial v_i}{\partial x_j} = \frac{\int_\Omega \frac{\partial v_i}{\partial n_j} d\Omega}{\int_\Omega d\Omega} = \frac{\int_S v_i n_j dS}{\int_\Omega d\Omega} = \frac{\sum_f v_i^f n_j^f \Delta S^f}{\Omega^O}$$

EURO Greece

Discretization of diffusion terms (1)

$$R_i^v = \frac{\partial (v_j v_i)}{\partial x_j} - \boxed{\frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]} + \frac{\partial p}{\partial x_i} = 0$$



$$-\int_\Omega \frac{\partial}{\partial x_j}\left[\nu_{Eff}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] d\Omega = -\int_S \left[\nu_{Eff}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] n_j dS =$$

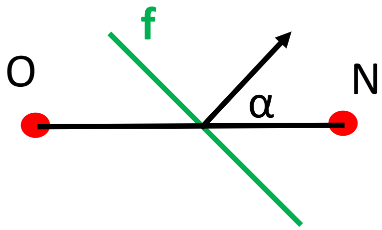
$$\underbrace{-\int_S \nu_{Eff}\frac{\partial v_i}{\partial x_j} n_j dS}_{D1} \underbrace{-\int_S \nu_{Eff}\frac{\partial v_j}{\partial x_i} n_j dS}_{D2}$$

$$D1 = -\int_S \nu_{Eff}\frac{\partial v_i}{\partial x_j} n_j dS = -\sum_f \nu_{Eff}^f \left(\frac{\partial v_i}{\partial x_j} n_j\right)^f \Delta S^f$$

$$\underbrace{\left(\frac{\partial v_i}{\partial x_j} n_j\right)^f = \frac{v_i^N - v_i^O}{\Delta^{NO}}}$$

linear

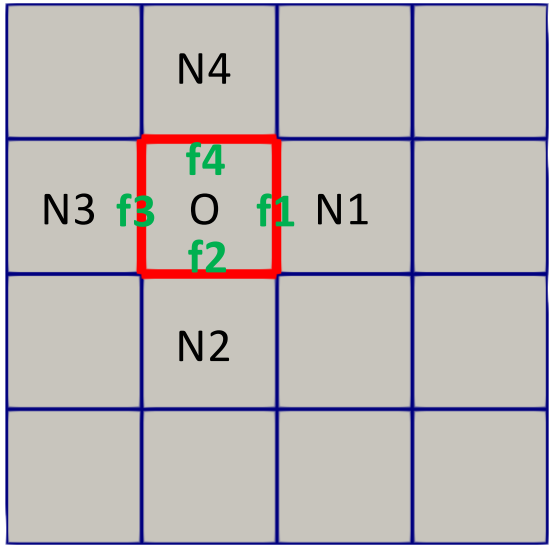
uncorrected. Assumes zero (or non-significant) non-orthogonality



Angle α: defines the non orthogonality of each faces → checkMesh

- **Correction for non-orthogonality:** `corrected` **surface normal Gradient (snGrad) scheme**
- **On meshes with very high non-orthogonality:** `limited 0.3333` **snGrad scheme**

Additional discretization for the momentum diffusion term



$$R_i^v = \frac{\partial (v_j v_i)}{\partial x_j} - \boxed{\frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]} + \frac{\partial p}{\partial x_i} = 0$$

$$-\int_\Omega \frac{\partial}{\partial x_j}\left[\nu_{Eff}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] d\Omega = -\int_S \left[\nu_{Eff}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] n_j dS =$$

$$\underbrace{-\int_S \nu_{Eff}\frac{\partial v_i}{\partial x_j} n_j dS}_{D1} \underbrace{- \int_S \nu_{Eff}\frac{\partial v_j}{\partial x_i} n_j dS}_{D2}$$

$$D2 = -\int_S \nu_{Eff}\frac{\partial v_j}{\partial x_i} n_j dS = -\sum_f \nu_{Eff}^f \left(\frac{\partial v_j}{\partial x_i}\right)^f n_j^f \Delta S^f$$

$$\left(\frac{\partial v_j}{\partial x_i}\right)^f = w^f \left.\frac{\partial v_j}{\partial x_i}\right|^O + (1 - w^f) \left.\frac{\partial v_j}{\partial x_i}\right|^N$$

Linear interpolation

Run with a first order scheme and compare

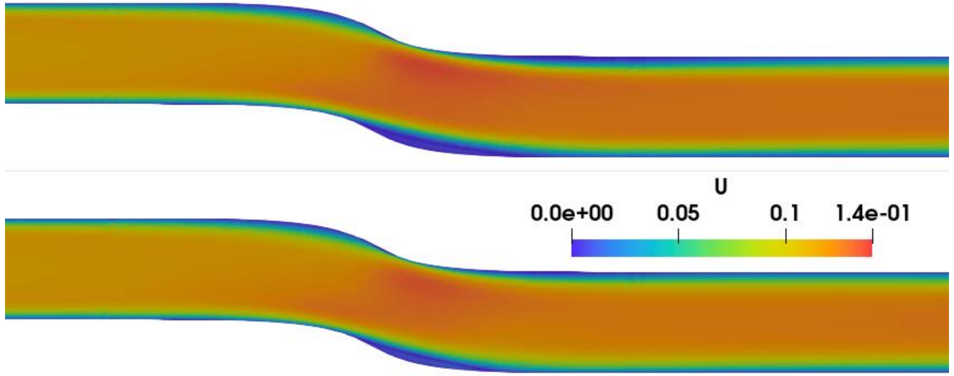
```
>> cd 2024_06_OF_training_EuroCC_Greece/02-firstOrder
```

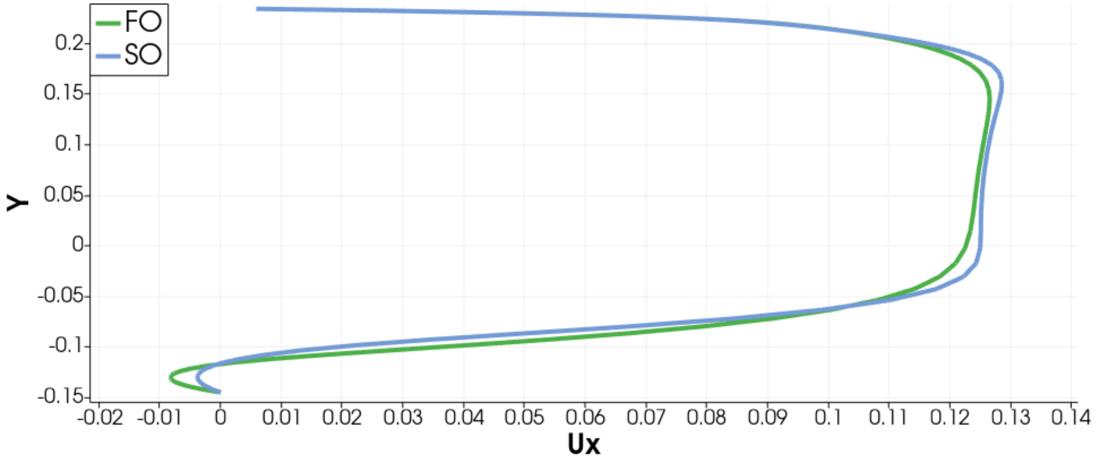
- Main change in `system/fvSchemes`

- See differences with the second order run by executing
```
>> diff system/fvSchemes ../01-secondOrder/system/fvSchemes
```

- Run the case and compare with the results from the second order run (total pressure losses, forces, etc)

- Compare the flow fields of the two runs in Paraview





Using *functions* through controlDict (1)

Entries in `system.functions` can be used:

- **To execute auxiliary code at the end of each iteration and/or run**
- **To compute new fields for post-processing (e.g. total pressure)**
- **To get useful post-processing content to files (e.g. residual history, flow rate, etc)**
- **Basic controls/entries for `functions` can be found [here](#)**
- **Useful examples of already setup functions can be found under `$FOAM_ETC/caseDicts/postProcessing`**

EURO Greece

Using *functions* through controlDict (2)

As an exercise, compute the forces on the *lower* and *upper* part of the duct

- Locate `forces` in `$FOAM_ETC/caseDicts/postProcessing`
- Copy a setup for incompressible flows and put it under `system`
- Replace the `patches` list with the one relevant to our case
- Include the additional function to `controlDict.functions`
- Execute only this function without re-running, by using
  `>> mpirun –np 4 simpleFoam –parallel –postProcess –func “forces” –latestTime`

When in doubt about the possible flags available to an executable, use the –help option, e.g
`>> simpleFoam -help`

Using *functions* through controlDict (3)

- **Run the case with first and second order accuracy for the convection terms (see comments for the `upwind` and `linearUpwind` schemes in slide 13)**

- **Compute the difference of pt losses between the Inlet and Outlet patches for the two cases**

The ARIS HPC system

- The Greek national HPC system
- Documentation in https://doc.aris.grnet.gr
- Multiple partitions
  - Thin: 8520 CPU cores
  - GPU: 88 NVidia K40 GPUs
  - ML: 8 NVidia V100 GPUs
  - FAT: CPU nodes with an emphasis on abundance of memory
- Apply for access: https://www.hpc.grnet.gr/access_policy
- Instructions on how to submit a job: https://doc.aris.grnet.gr/run

960 CPUs running for 24h = 23040 CPUhours