

# A practical introduction to OpenFOAM - Theory background and first steps

7<sup>th</sup> August 2023

Stefano Lorenzi – Politecnico di Milano ([stefano.lorenzi@polimi.it](mailto:stefano.lorenzi@polimi.it))

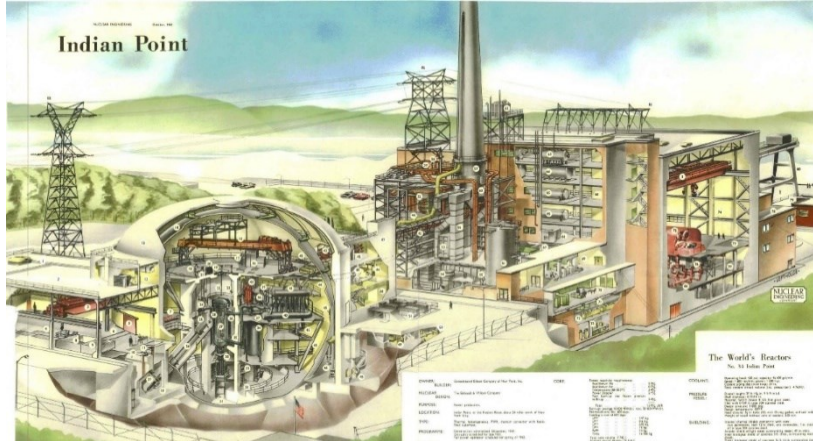
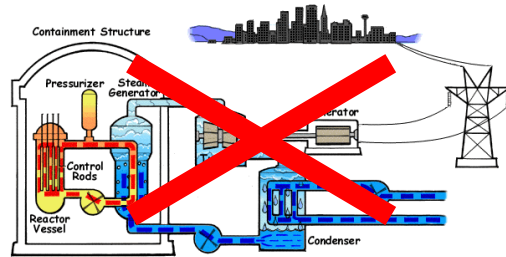
# Objective of the lecture

## We will have a look on:

- “Multi-physics modelling and simulation”
- How to model a nuclear engineering problem: from the physical system to the mathematical and numerical modeling to result analysis
- Overview on PDEs, Finite Volume discretization method and linear algebra solvers in the view of OF use
- Solver and case folder structure (where to find equations, parameters, boundary conditions, and initial conditions),
- First steps with OF

# Modelling and Simulation

## Nuclear systems are complex systems



Multicomponent

Multidisciplinary

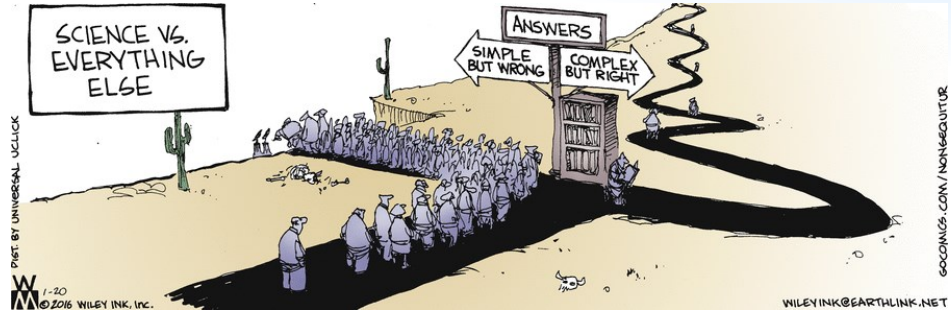
Multiphysics

Multiscale

# Modelling and Simulation

## How to study nuclear reactor (or complex system)?

“For every complex problem there is an answer that is clear, simple, and... wrong” (H. L. Mencken)

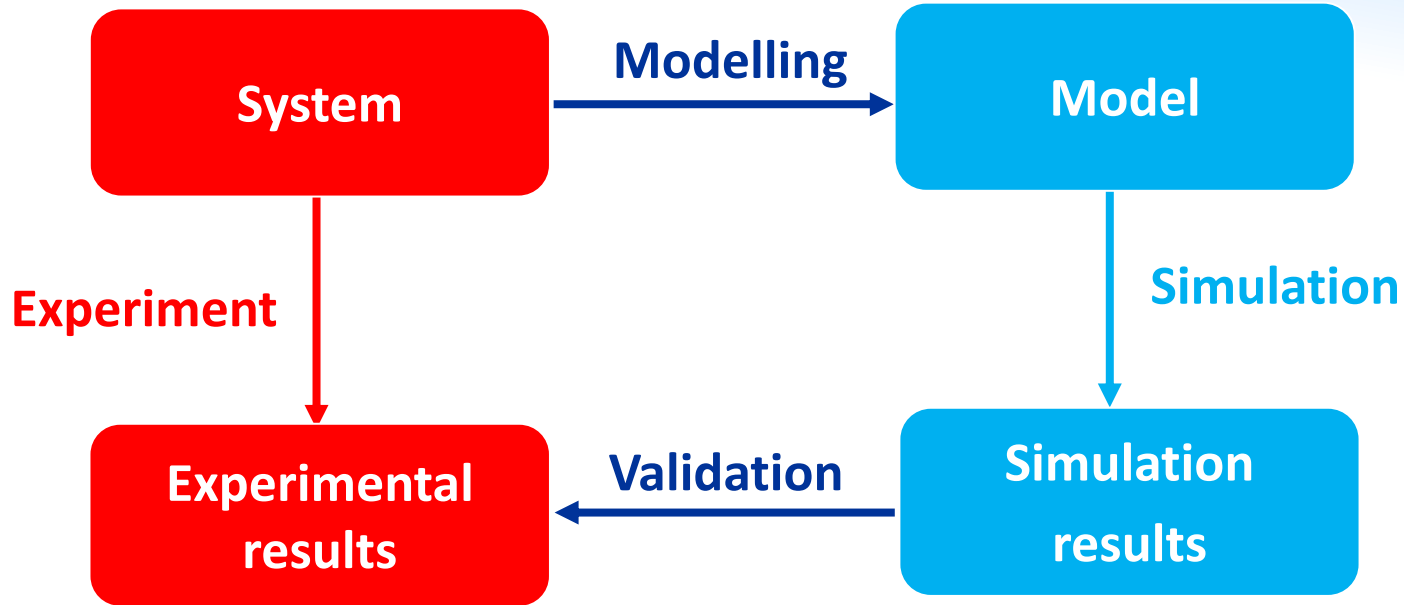


“There are no simple answers to complex problems”

**Proper analysis:** The analysis performed for a system should not be more complex than is necessary for a given purpose

- . We should know which is our purpose (what we want to know about the problem)
- . We should know which are the limits of our analysis approach (what we can studied with this approach or not)

# Modelling and Simulation



# Modelling and Simulation

## **To design and analyze system (prediction of plant parameters)**

- Increase understanding
- Optimization
- Control
- Safety

## **Advantages**

- Experiment are expensive, dangerous and cannot be performed on system that cannot exist
- Time scale
- Variables may be inaccessible
- Easy manipulation
- No disturbances
- Exploring different scenario

# Modelling and Simulation

## Critical issues

- A model is not the real world – Pygamlion effect
- Forcing reality into the constraints of a model – Procrustes effect
- Forgetting the model's level of accuracy
- Forgetting the purpose of the model – **Proper modelling**
- The computational resources are not unlimited



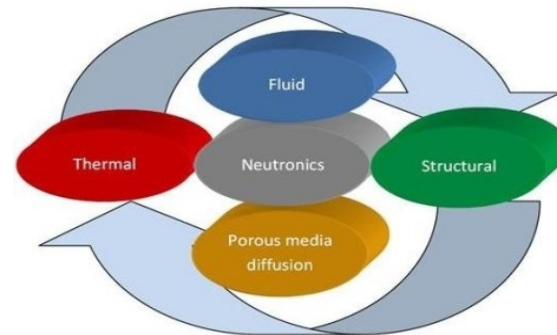
# Multiphysics

“Multiphysics is a **computational discipline** which treats simulations that involve multiple physical models or multiple simultaneous physical phenomena” (Wikipedia et al., 2017)



**“Multiphysics is the study of the mutual interaction of different physical phenomena”**

- Fluid dynamics
- Heat Transfer
- Neutronics
- Chemistry
- Mechanics
- BoP





# Modelling of an engineering problem

**“How do you model a nuclear engineering problem?”**

**“Which are the steps required to model a nuclear engineering problem?”**

# Modelling of an engineering problem

**“Which are the step required to model a nuclear engineering problem?”**

- 1. Physical modelling:** choose the proper modelling approach (PDEs)
- 2. Domain modelling:** establish the boundary of the system and create the geometry (with possible assumptions)
- 3. Domain + equation discretization:** selection of a discretization method (FV, FE, DG,...) + mesh creation + equation discretization (including selection of boundary and initial conditions)
- 4. Solution of (discretized) equations:** selection of the method to solve the system of linear algebraic equations
- 5. Post-processing:** visualization of the results

# Modelling of an engineering problem



**“Which are the step required to model a nuclear engineering problem?”**

- 1. Physical modelling:** choose the proper modelling approach (PDEs)
- 2. Domain modelling:** establish the boundary of the system and create the geometry (with possible assumptions)
- 3. Domain + equation discretization:** selection of a discretization method (FV, FE, DG,...) + mesh creation + equation discretization (including selection of boundary and initial conditions)
- 4. Solution of (discretized) equations:** selection of the method to solve the system of linear algebraic equations
- 5. Post-processing:** visualization of the results

**Solver** vs **Simulation case**

# Modelling of an engineering problem

“Which are the step required to model a nuclear engineering problem?”

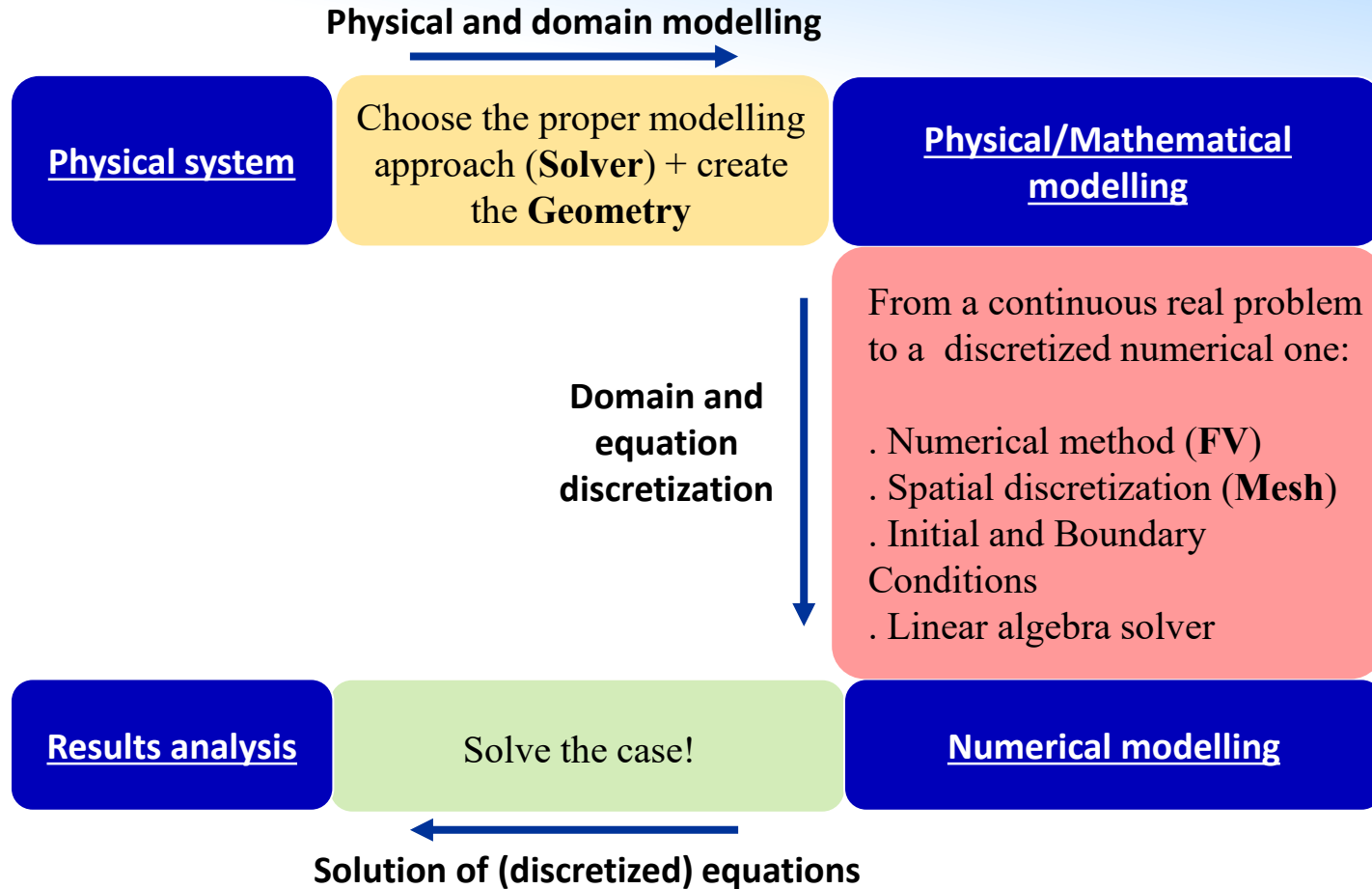
1. Physical modelling
2. Domain modelling
3. Domain + equation discretization
4. Solution of (discretized) equations
5. Post-processing

## OpenFOAM Workflow

Workflow mirrors that of traditional CFD workflow

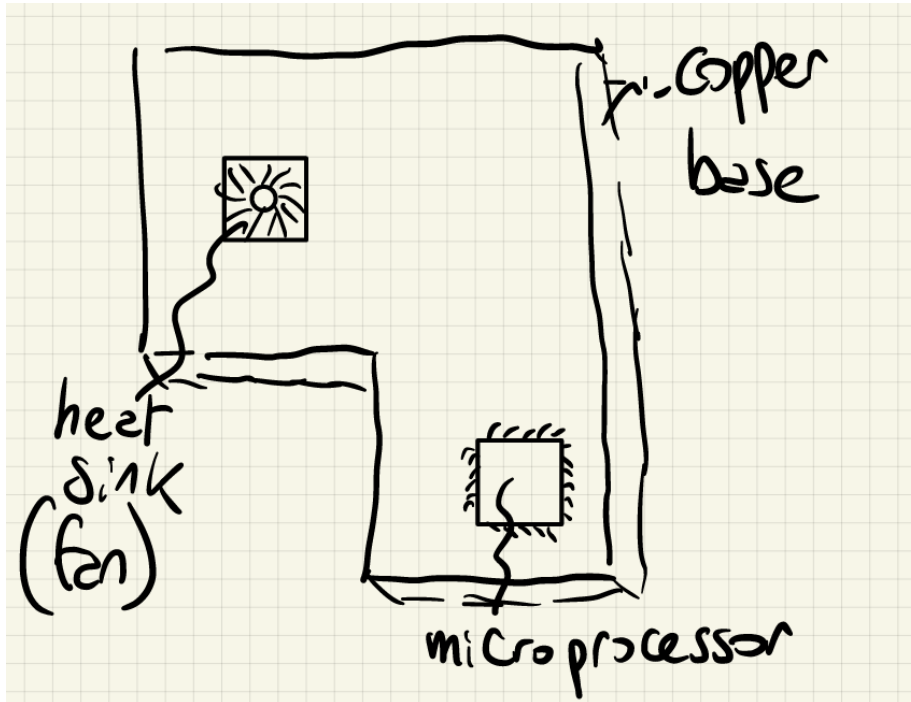


# Modelling of an engineering problem



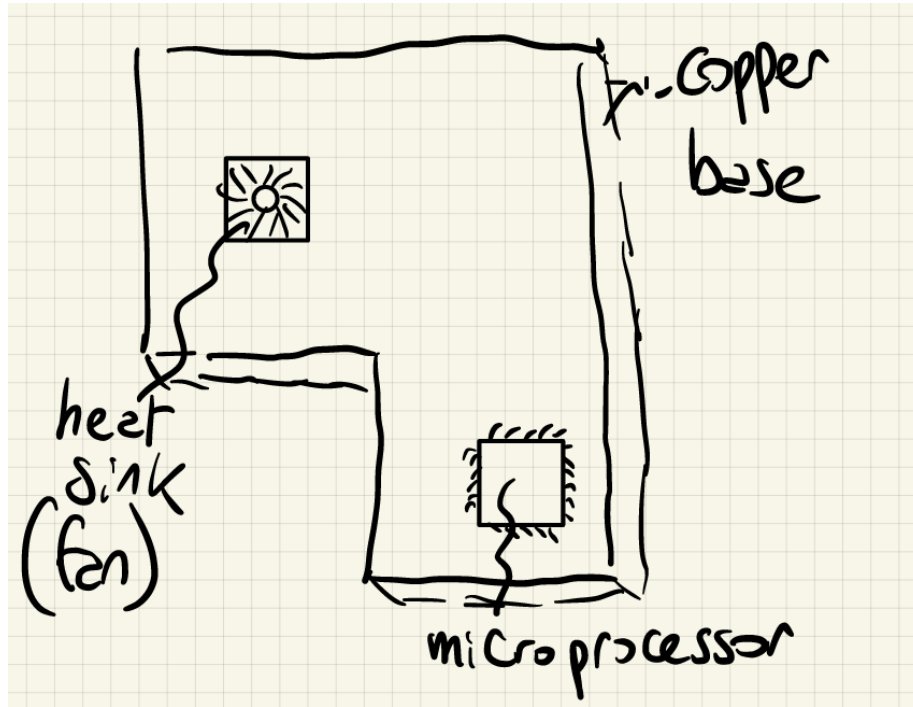
# Modelling of an engineering problem

Example taken from «F. Moukalled, L. Mangani, M. Darwish. The Finite Volume Method in Computational Fluid Dynamics. Springer, 2016»



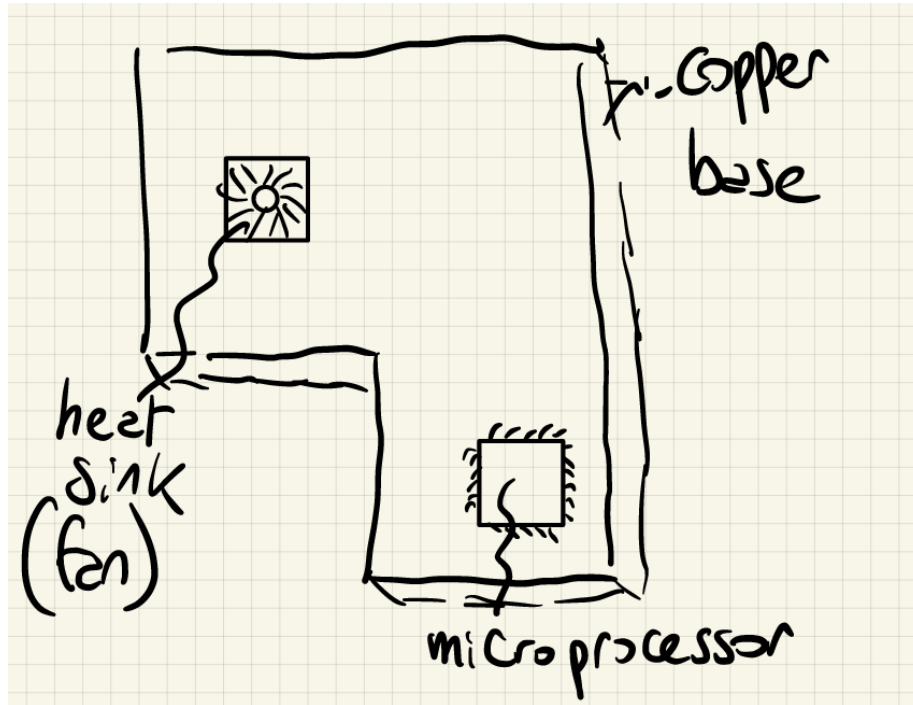
**Problem:** study the heat transfer in a electronic device where a microprocessor is connected to a heat sink

# Modelling of an engineering problem



1. Physical modelling: **how do we model the heat transfer?**

# Modelling of an engineering problem



**1. Physical modelling:** how do we model the heat transfer?

$$-\nabla \cdot (k\nabla T) = \dot{q}$$

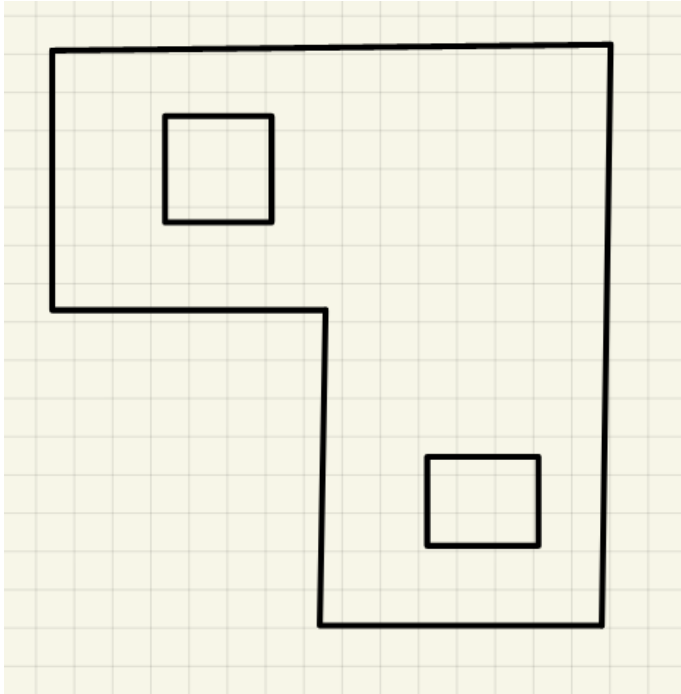
In OF, the physical model is decided by the **solver** you are using!

You can find the available solvers in  
[/OpenFOAM-XXX/applications/solvers/](#)

**Which could be a solver for our problem?**



# Modelling of an engineering problem

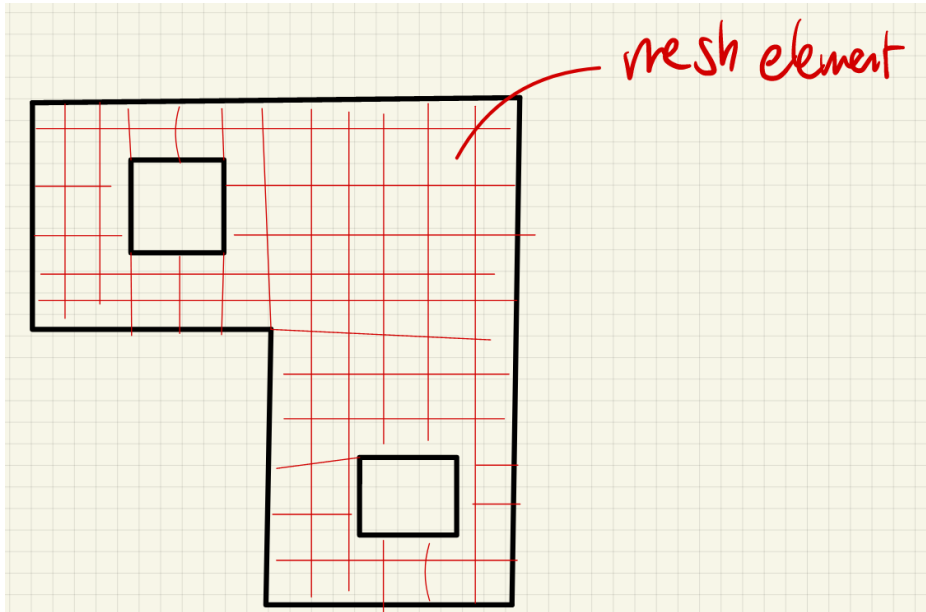


**2. Domain modelling:** we need to approximate the real geometry.

This is the first step of the **simulation of a case**. To build the geometry, you can use custom or external utilities/programmes.

A simulation in OF (case) follows a specific file structure.

# Modelling of an engineering problem

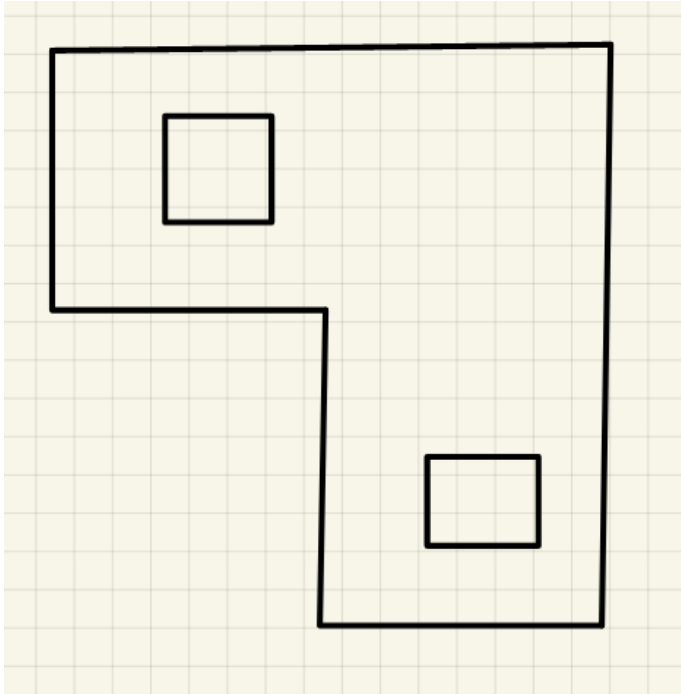


**3. Domain discretization:** It is required since analytical solution of PDE is usually difficult (impossible to achieve) → we need an approximation

In OF, the discretization method is the Finite Volume (FV) method.

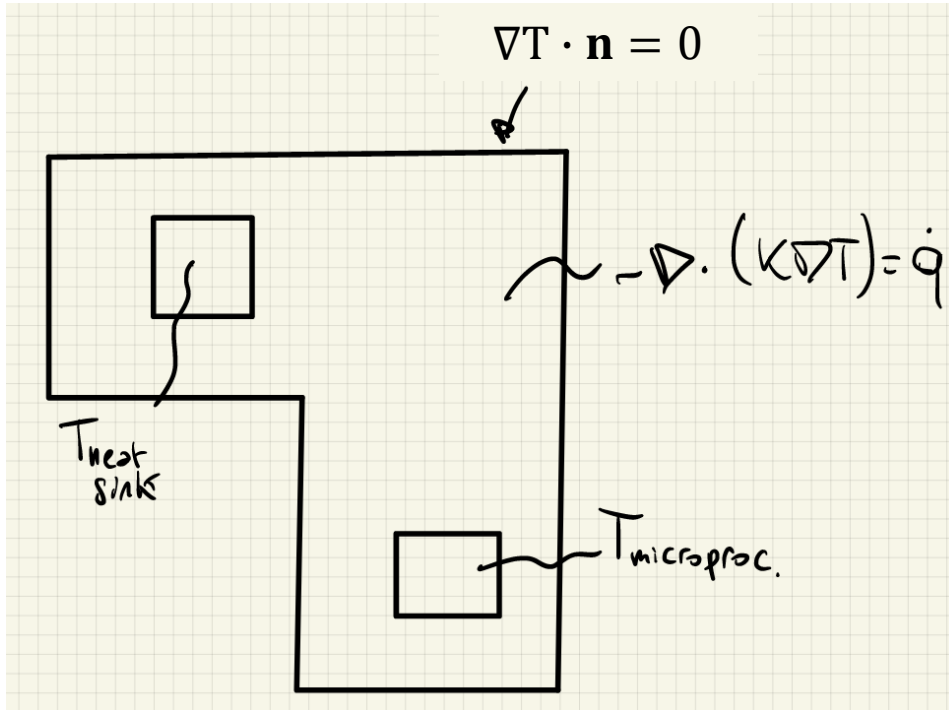
Once created the geometry, we need a **mesh** on which the PDE is solved.

# Modelling of an engineering problem



**3. Equation discretization:** Thanks to the FVM we transform the PDE in a discretized form. We need to specify **the boundary conditions, initial conditions plus the discretization options** (aka discretization schemes)

# Modelling of an engineering problem



**3. Equation discretization:** Thanks to the FVM we transform the PDE in a discretized form. We need to specify **the boundary conditions, initial conditions plus the discretization options** (aka discretization schemes)

$$\frac{\partial (\rho \phi)}{\partial t} + \nabla \cdot (\rho \vec{v} \phi) = \nabla \cdot (\Gamma \nabla \phi) + Q$$

↓ DISCRETIZATION

$$\partial_c \phi_c + \sum_{F \in \text{NB}(c)} \partial_f \phi_f = b_c$$

# Modelling of an engineering problem

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho \vec{v} \phi) = \nabla \cdot (\Gamma \nabla \phi) + Q$$

⇓ DISCRETIZATION

$$\rho_c \phi_c + \sum_{F \in \text{NB}(c)} a_f \phi_f = b_c$$

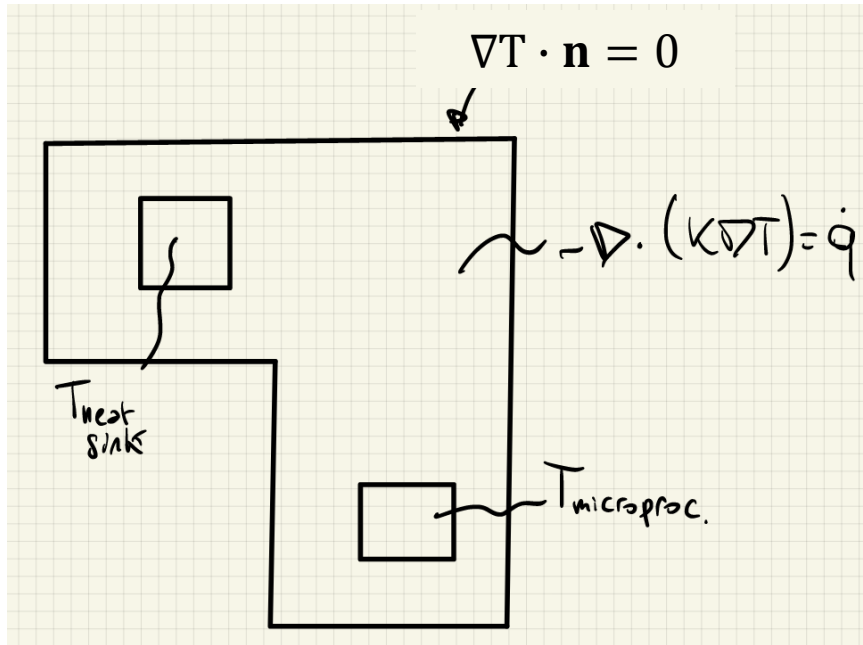
⇓

$$\underline{A} \underline{x} = \underline{b} \quad \text{LINEAR SYSTEM}$$

**4. Solution of (discretized) equations:** The discretized form is nothing more than a system of linear algebraic equations.

We need to specify **how to solve this linear system** (iterative or direct solvers?)

# Modelling of an engineering problem



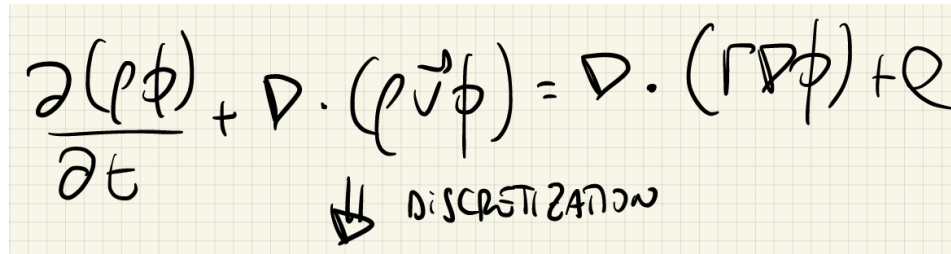
This is an example involving one physics. In case of a multiphysics system the mathematical modelling and the numerical consideration are usually interconnected.

In a multiphysics solver, we need to specify how do we solve the multiphysics coupling.

In OF, this is usually done with operator-splitting (one equation at time + fixed point iteration)

# Introduction on FV

The purpose of any discretisation practice is to transform one or more partial differential equations into a corresponding system of algebraic equations.


$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho \vec{u} \phi) = \nabla \cdot (\Gamma \nabla \phi) + Q$$

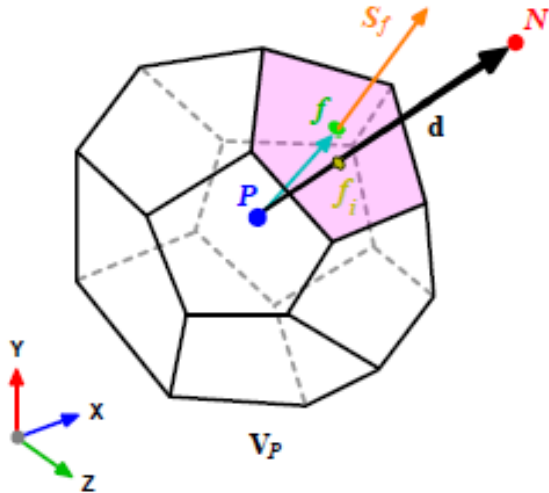
↓ DISCRETIZATION

$$\int_{V_P} \underbrace{\frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \int_{V_P} \underbrace{\nabla \cdot (\rho \mathbf{u} \phi) dV}_{\text{convective term}} - \int_{V_P} \underbrace{\nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV}_{\text{diffusion term}} = \int_{V_P} \underbrace{S_\phi(\phi) dV}_{\text{source term}}$$

# Introduction on FV

Two ingredients: **domain discretization** and equation discretization

The domain is divided in control volumes  $V_p$  where the solution is calculated by integrating the PDE over  $V_p$ .



- .  $P$  is the centroid of the corresponding CV
- .  $N$  is the centroid of the neighbor CV
- .  $d$  is the distance from  $P$  to  $N$
- .  $f$  is the control volume face
- .  $S_f$  is the face area vector pointing outwards from the control volume, normal to the face with magnitude equal to the area of the face



# Introduction on FV

- Two ingredients: domain discretization and **equation discretization**

$$\underbrace{\int_{V_P} \frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \underbrace{\int_{V_P} \nabla \cdot (\rho \mathbf{u} \phi) dV}_{\text{convective term}} - \underbrace{\int_{V_P} \nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV}_{\text{diffusion term}} = \underbrace{\int_{V_P} S_\phi(\phi) dV}_{\text{source term}}$$

H<sub>p</sub>: linear variation in space and time

$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla \phi)_P \quad \text{where} \quad \phi_P = \phi(\mathbf{x}_P)$$
$$\phi(t + \delta t) = \phi^t + \delta t \left( \frac{\partial \phi}{\partial t} \right)^t \quad \text{where} \quad \phi^t = \phi(t)$$

Gauss (divergence) theorem to convert the volume integrals into surface integrals.

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$

# Introduction on FV

$$\int_{V_P} \underbrace{\frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \int_{V_P} \underbrace{\nabla \cdot (\rho \mathbf{u} \phi) dV}_{\text{convective term}} - \int_{V_P} \underbrace{\nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV}_{\text{diffusion term}} = \int_{V_P} \underbrace{S_\phi(\phi) dV}_{\text{source term}}$$

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$

$$\frac{\partial}{\partial t} \int_{V_P} (\rho \phi) dV + \oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho \mathbf{u} \phi)}_{\text{convective flux}} - \oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho \Gamma_\phi \nabla \phi)}_{\text{diffusive flux}} = \int_{V_P} S_\phi(\phi) dV$$

$$\oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho \Gamma_\phi \nabla \phi)}_{\text{diffusive flux}} = \sum_f \int_f d\mathbf{S} \cdot (\rho \Gamma_\phi \nabla \phi)_f = \sum_f \mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f$$

$$\int_{V_P} \underbrace{\frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \sum_f \underbrace{\mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f}_{\text{convective flux}} - \sum_f \underbrace{\mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f}_{\text{diffusive flux}} = \underbrace{(S_c V_P + S_p V_P \phi_P)}_{\text{source term}}$$

# Introduction on FV

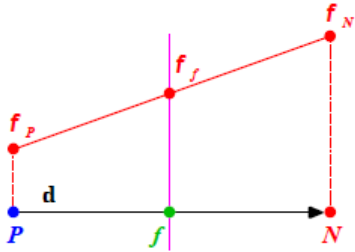
$$\int_{V_P} \underbrace{\frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \sum_f \underbrace{\mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f}_{\text{convective flux}} - \sum_f \underbrace{\mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f}_{\text{diffusive flux}} = \underbrace{(S_c V_P + S_p V_P \phi_P)}_{\text{source term}}$$

. In the convective and diffusive flux, the variable of interest is calculated at the face flux but we want the value at the centroid **P**

. The face values appearing in the convective and diffusive fluxes have to be computed by some form of interpolation from the centroid values of the control volumes at both sides of face *f*.

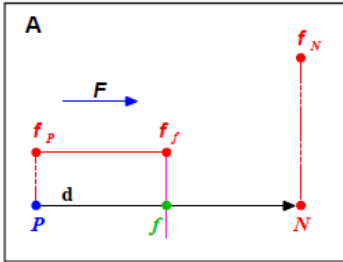
# Introduction on FV

## Convective flux



$$\phi_f = f_x \phi_P + (1 - f_x) \phi_N$$

$$f_x = \frac{fN}{PN} = \frac{|\mathbf{x}_f - \mathbf{x}_N|}{|\mathbf{d}|}$$



$$\phi_f = \begin{cases} \phi_f = \phi_P & \text{for } \dot{F} \geq 0, \\ \phi_f = \phi_N & \text{for } \dot{F} < 0. \end{cases}$$

## Linear interpolation

(central differencing)

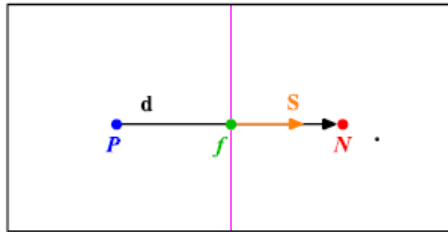
- . 2nd order accurate
- . May create oscillatory solution

## Upwind differencing

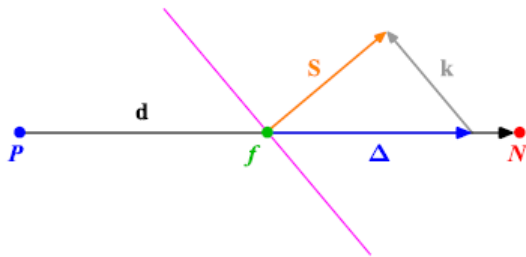
- . 1st order accurate
- . Bounded solution
- . Diffusive

# Introduction on FV

## Diffusive flux



$$\mathbf{S} \cdot (\nabla \phi)_f = |\mathbf{S}| \frac{\phi_N - \phi_P}{|\mathbf{d}|}$$



$$\mathbf{S} = \Delta_{\perp} + \mathbf{k}$$

$$\mathbf{S} \cdot (\nabla \phi)_f = \underbrace{|\Delta_{\perp}| \frac{\phi_N - \phi_P}{|\mathbf{d}|}}_{\text{orthogonal contribution}}$$

$$+ \underbrace{\mathbf{k} \cdot (\nabla \phi)_f}_{\text{non-orthogonal contribution}}$$

## Linear interpolation (central differencing)

. II order accurate

. Truncation error

introduced due to the  
mesh non-orthogonality

# Introduction on FV

. Last but not least.. Time discretization

$$\int_t^{t+\Delta t} \left[ \left( \frac{\partial \rho \phi}{\partial t} \right)_P V_P + \sum_f \mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f - \sum_f \mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f \right] dt$$
$$= \int_t^{t+\Delta t} (S_c V_P + S_p V_P \phi_P) dt.$$

$$\left( \frac{\partial \rho \phi}{\partial t} \right)_P = \frac{\rho_P^n \phi_P^n - \rho_P^o \phi_P^o}{\Delta t}, \quad \phi^n = \phi(t + \Delta t),$$
$$\int_t^{t+\Delta t} \phi(t) dt = \frac{1}{2}(\phi^o + \phi^n) \Delta t, \quad \phi^o = \phi(t).$$

- .  $\phi^0$  is the value at the previous time step (known)
- .  $\phi^n$  is the value at the current time step (unknown)

# Introduction on FV

$$\phi_f = f_x \phi_P^o + (1 - f_x) \phi_N^o,$$

$$\mathbf{S} \cdot (\nabla \phi)_f = |\Delta| \frac{\phi_N^o - \phi_P^o}{|d|} + \mathbf{k} \cdot (\nabla \phi)_f.$$

$$\phi_f = f_x \phi_P^n + (1 - f_x) \phi_N^n,$$

$$\mathbf{S} \cdot (\nabla \phi)_f = |\Delta| \frac{\phi_N^n - \phi_P^n}{|d|} + \mathbf{k} \cdot (\nabla \phi)_f.$$

$$\begin{aligned} \frac{\rho_P \phi_P^n - \rho_P \phi_P^o}{\Delta t} V_P + \frac{1}{2} \sum_f F \phi_f^n - \frac{1}{2} \sum_f (\rho \Gamma_\phi)_f \mathbf{S} \cdot (\nabla \phi)_f^n \\ + \frac{1}{2} \sum_f F \phi_f^o - \frac{1}{2} \sum_f (\rho \Gamma_\phi)_f \mathbf{S} \cdot (\nabla \phi)_f^o \\ = Su V_P + \frac{1}{2} Sp V_P \phi_P^n + \frac{1}{2} Sp V_P \phi_P^o. \end{aligned}$$

## Euler explicit (very bad)

- . 1 order accurate
- . Possibly unstable

## Euler implicit

- . 1 order accurate
- . Stable
- . Leads to a linear system

## Crank Nicholson

- . 2 order accurate
- . Stable
- . Leads to a linear system





# Modelling of an engineering problem in OF

**“Which are the step required to model a nuclear engineering problem?”**

1. Physical modelling
2. Domain modelling
3. Domain discretization
4. Solution of (discretized) equations
5. Post-processing

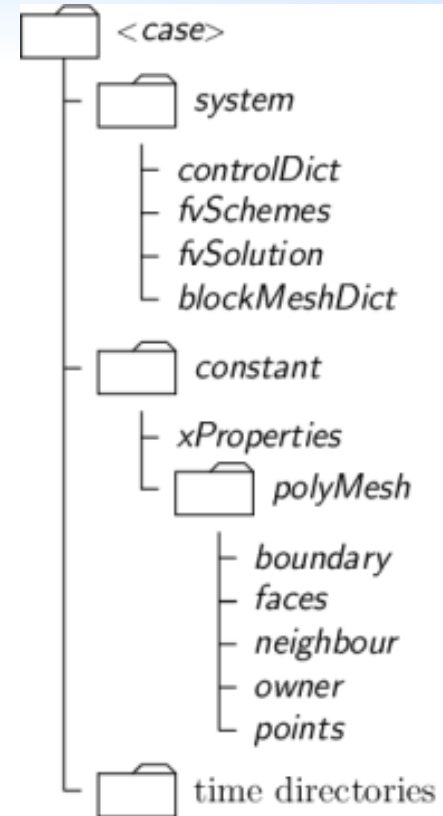
→ Solver



# OpenFOAM case

## How to perform simulation with OpenFOAM:

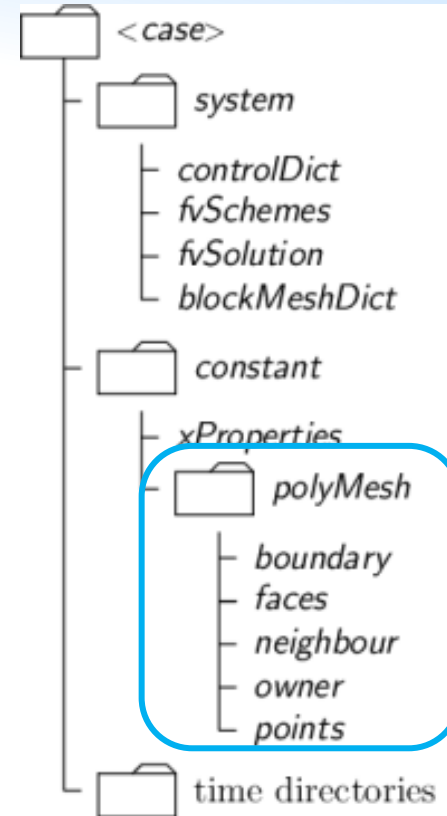
- 1) Select the solver, i.e., the modelling
- 2) Case preparation
  - . Geometry and mesh
  - . Initial and boundary conditions
  - . Physical models/parameters
  - . Discretization
  - . Solver parameters
  - . Simulation parameters
- 3) Post-processing
  - . Visualization
  - . Data analysis



# OpenFOAM case

## 2) Case preparation

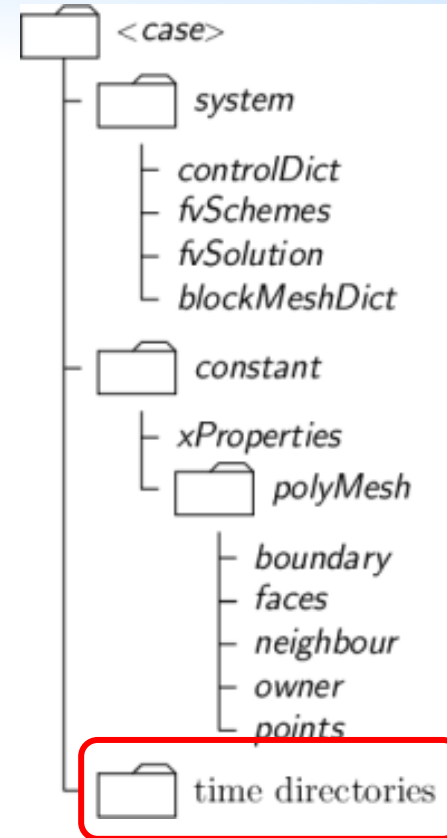
- . **Geometry and Mesh**
- . Initial and boundary conditions
- . Physical models/parameters
- . Discretization
- . Solver parameters
- . Simulation parameters



# OpenFOAM case

## 2) Case preparation

- . Geometry and Mesh
- . **Initial and boundary conditions**
- . Physical models/parameters
- . Discretization
- . Solver parameters
- . Simulation parameters



# OpenFOAM case

Initial  
conditions



```
17 dimensions [0 1 -1 0 0 0];
18
19 internalField uniform (0 0 0);
20
21 boundaryField
22 {
23     movingWall
24     {
25         type    fixedValue;
26         value    uniform (1 0 0);
27     }
28
29     fixedWalls
30     {
31         type    fixedValue;
32         value    uniform (0 0 0);
33     }
34
35     frontAndBack
36     {
37         type    empty;
38     }
39 }
```

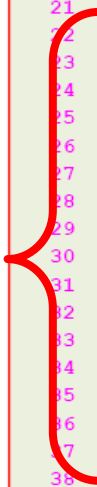
Dimensions of the field  $\frac{m}{s}$

Numerical boundary condition for the patch **movingWall**

Numerical boundary condition for the patch **fixedWalls**

Numerical boundary condition for the patch **frontAndBack** (this is a constrained boundary condition).

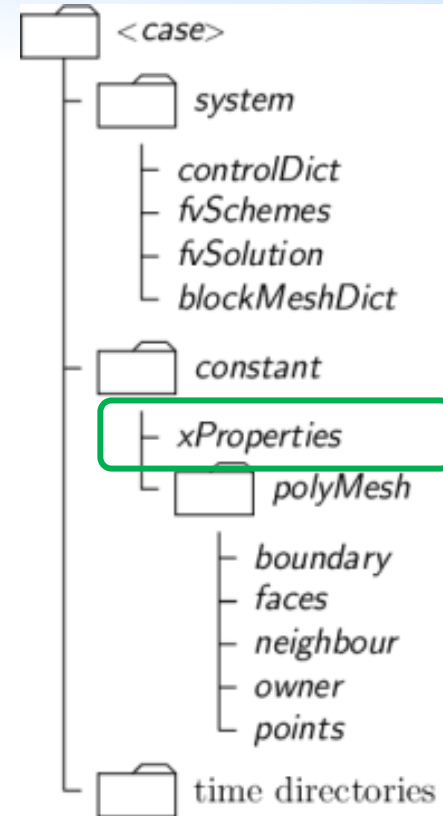
Boundary  
conditions



# OpenFOAM case

## 2) Case preparation

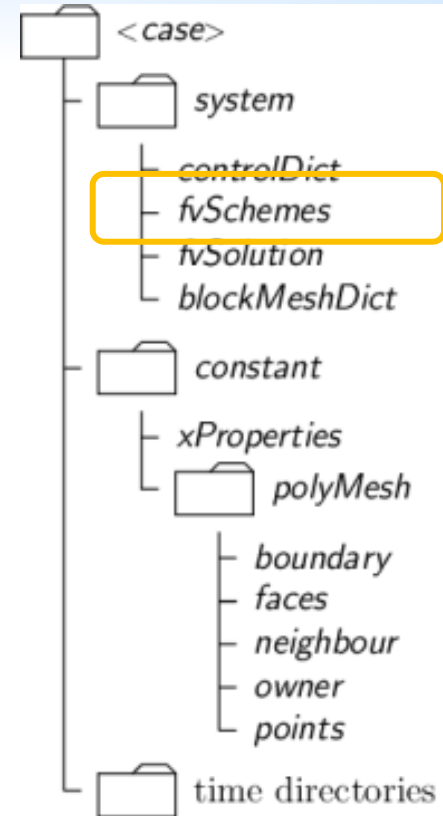
- . Geometry and Mesh
- . Initial and boundary conditions
- . **Physical models/parameters**
- . Discretization
- . Solver parameters
- . Simulation parameters



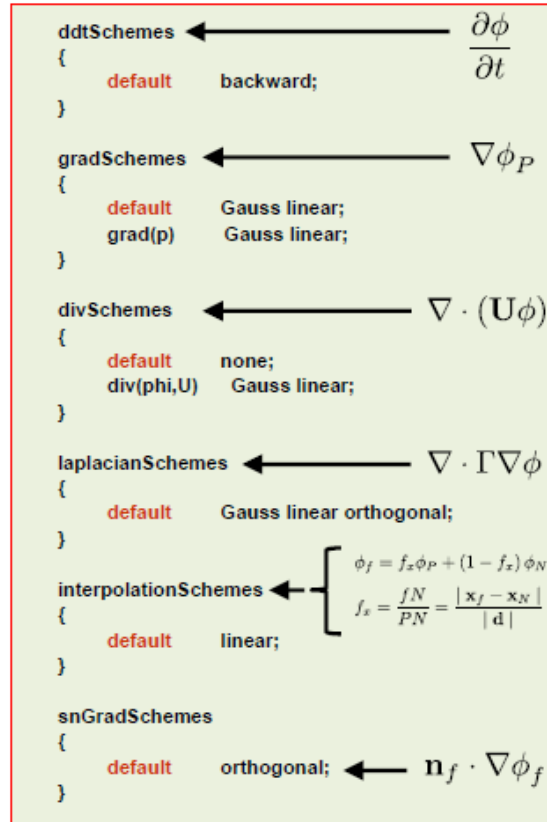
# OpenFOAM case

## 2) Case preparation

- . Geometry and Mesh
- . Initial and boundary conditions
- . Physical models/parameters
- . **Discretization**
- . Solver parameters
- . Simulation parameters



# OpenFOAM case

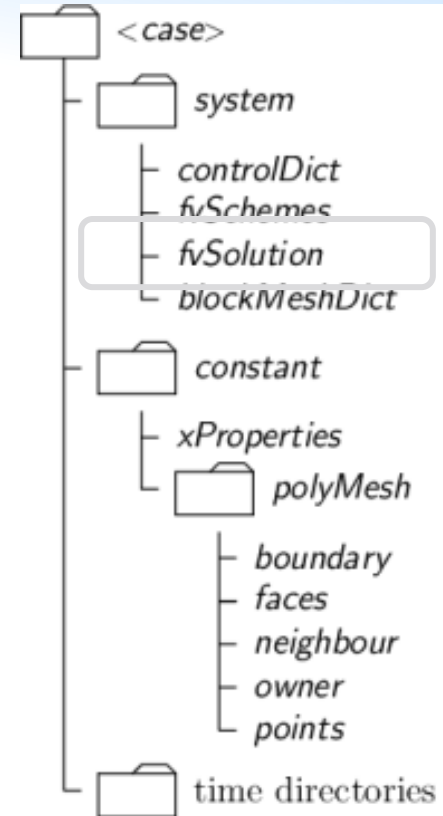




# OpenFOAM case

## 2) Case preparation

- . Geometry and Mesh
- . Initial and boundary conditions
- . Physical models/parameters
- . Discretization
- . Solver parameters
- . Simulation parameters



# OpenFOAM case

. Info on the linear (iterative) solver for each discretized matrix-related quantity

+ Info on the under-relaxation

. Info related to the method for Navier – Stokes equation

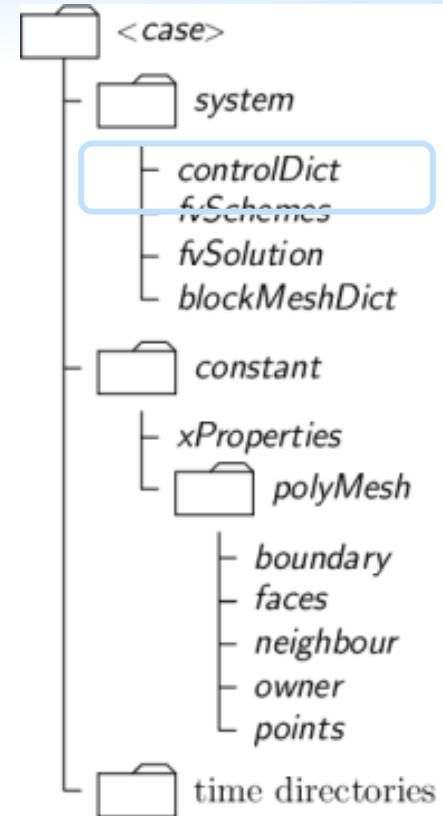
```
solvers ←  
{  
  p  
  {  
    solver      PCG;  
    preconditioner DIC;  
    tolerance   1e-06;  
    relTol      0;  
  }  
  pFinal  
  {  
    $p;  
    relTol 0;  
  }  
  U  
  {  
    solver      PBiCGStab;  
    preconditioner DILU;  
    tolerance   1e-08;  
    relTol      0;  
  }  
}  
  
PISO ←  
{  
  nCorrectors 2;  
  nNonOrthogonalCorrectors 1;  
}
```

```
relaxationFactors  
{  
  fields  
  {  
    p 0.8;  
  }  
  equations  
  {  
    "(U|k|epsilon)" 0.6;  
  }  
}
```

# OpenFOAM case

## 2) Case preparation

- . Geometry and Mesh
- . Initial and boundary conditions
- . Physical models/parameters
- . Discretization
- . Solver parameters
- . **Simulation parameters**



# OpenFOAM case

```
startFrom latestTime;
startTime 0; ←
stopAt endTime;
endTime 10000; ←
deltaT 1; ←
writeControl runTime;
writeInterval 100; ←
purgeWrite 10; ←
writeFormat ascii;
writePrecision 8;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable yes; ←
```

# Modelling of an engineering problem in OF

**“Which are the step required to model a nuclear engineering problem?”**

1. Physical modelling
2. Domain modelling
3. Domain discretization
4. Solution of (discretized) equations
5. Post-processing

→ Solver



# OpenFOAM solver

Let's take a laminar fluid dynamics problem...

Physical modelling = Incompressible Navier – Stokes equations

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla p / \rho_0 + \mathbf{F} \end{cases}$$

Which is the structure of an hypothetical OF solver for Incompressible NS?

In *OpenFOAM-XXX/applications/solvers/incompressible/pimpleFoam*

**Make**

pimpleFoam.C

Ueqn.H

correctPhi.H

createFields.H

pEqn.H

setRDeltaT.H

# OpenFOAM solver



```
pimpleFoam.C //Main file of the solver
```

```
#include "fvCFD.H"
#include "dynamicFvMesh.H"
#include "singlePhaseTransportModel.H"
#include "turbulentTransportModel.H"
#include "pimpleControl.H"
#include "CorrectPhi.H"
#include "fvOptions.H"
#include "localEulerDdtScheme.H"
#include "fvcSmooth.H"

// * * * * *
//

int main(int argc, char *argv[])
{
    argList::addNote
    (
        "Transient solver for incompressible, turbulent flow"
        " of Newtonian fluids on a moving mesh."
    );
}
```

# OpenFOAM solver

`pimpleFoam.C` //Main file of the solver



```
while (runTime.run())
{
    ++runTime;
    Info<< "Time = " << runTime.timeName() << nl << endl;
    // --- Pressure-velocity PIMPLE corrector loop
    while (pimple.loop())
    {
        #include "UEqn.H"
        // --- Pressure corrector loop
        while (pimple.correct())
        {
            #include "pEqn.H"
        }
        if (pimple.turbCorr())
        {
            laminarTransport.correct();
            turbulence->correct();
        }
    }
    runTime.write();
    runTime.printExecutionTime(Info);
}
Info<< "End\n" << endl;
```



```
MRF.correctBoundaryVelocity(U);
```

```
tmp<fvVectorMatrix> tUEqn
```

```
(  
    fvm::ddt(U) + fvm::div(phi, U)  
    + MRF.DDt(U)  
    + turbulence->divDevReff(U)  
    ==  
    fvOptions(U)
```

```
);  
fvVectorMatrix& UEqn = tUEqn.ref();
```

```
UEqn.relax();
```

```
fvOptions.constrain(UEqn);
```

```
if (pimple.momentumPredictor())  
{  
    solve(UEqn == -fvc::grad(p));  
  
    fvOptions.correct(U);  
}
```

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nu \Delta \mathbf{u} + \nabla p = 0$$

## Equation Mimicking

OpenFOAM 

*The Open Source CFD Toolbox*

- Natural language of continuum mechanics: **partial differential equations**
- Example: turbulence kinetic energy equation

$$\frac{dk}{dt} + \nabla \cdot (\bar{u}k) - \nabla \cdot [(v + \nu_t)\nabla k] = \nu_t \left[ \frac{1}{2} (\nabla \bar{u} + \nabla \bar{u}^T) \right]^2 - \frac{\epsilon_0}{k_0} k$$

- Objective: represent PDEs in their natural language

```
solve  
(  
    fvm::ddt(k)  
    + fvm::div(phi, k)  
    - fvm::laplacian(nu() + nut, k)  
    ==  
    nut*magSqr(symm(fvc::grad(U)))  
    - fvm::Sp(epsilon/k, k)  
);
```

- Correspondence between implementation and equation is clear

# OpenFOAM solver

pEqn.H



```
volScalarField rAU(1.0/UEqn.A());
volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));
surfaceScalarField phiHbyA("phiHbyA", fvc::flux(HbyA));
if (pimple.consistent())
{
    rAtU = 1.0/max(1.0/rAU - UEqn.H1(), 0.1/rAU);
    phiHbyA +=
        fvc::interpolate(rAtU() - rAU)*fvc::snGrad(p)*mesh.magSf();
    HbyA -= (rAU - rAtU()*fvc::grad(p));
}
// Non-orthogonal pressure corrector loop
while (pimple.correctNonOrthogonal())
{
    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAtU(), p) == fvc::div(phiHbyA)
    );

    if (pimple.finalNonOrthogonalIter())
    {
        phi = phiHbyA - pEqn.flux();
    }
}
```

# OpenFOAM solver

**Any remarks on the solver structure?**

# OpenFOAM solver

## Any remarks on the solver structure?

- . Four iteration loop (time loop, “pimple” loop, “pressure correction” loop and nonOrthogonal correction loop)
- . No straightforward way for the velocity – pressure coupling
- . Use of `fvm::` vs `fvc::` e.g. `fvm::ddt(U)`, `fvc::grad(p)`

# OpenFOAM solver

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla p / \rho_0 + \mathbf{F} \end{cases}$$

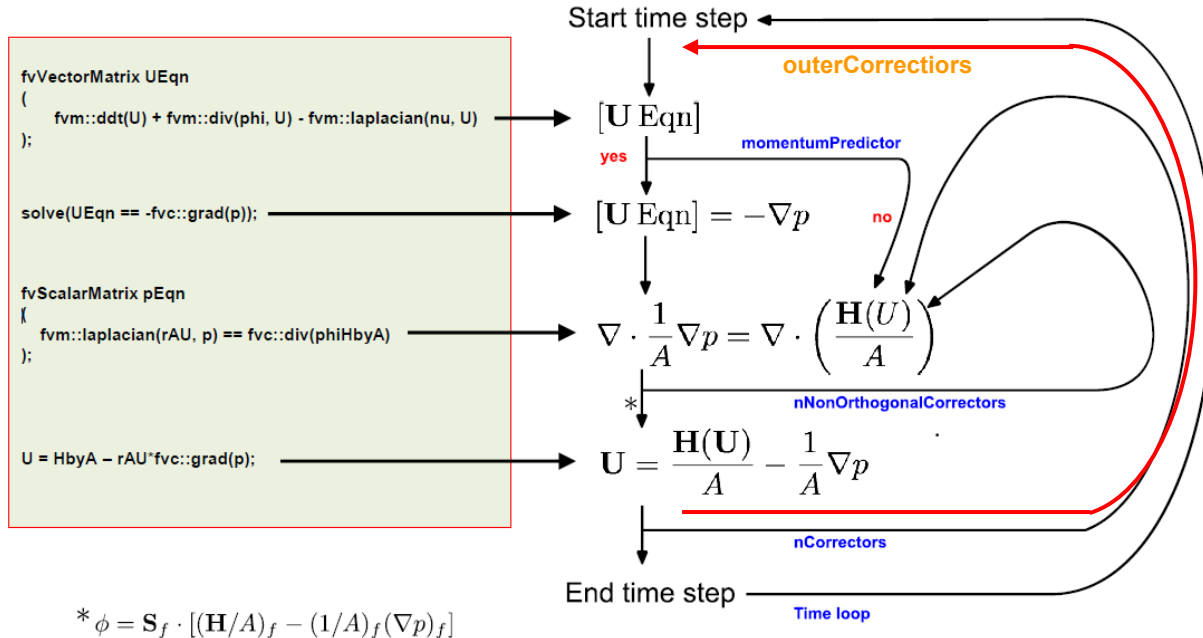
Navier – Stokes equations pose different numerical challenges:

- . Non linearity of the convective term  $(\mathbf{u} \cdot \nabla) \mathbf{u}$
- . There is no “pressure” equation (actually the continuity equation imposes a constraint on the momentum equation)

In OF, this is tackled with segregated pressure-based (predictor-corrector) methods as SIMPLE, PISO, PIMPLE where:

- .  $(\mathbf{u} \cdot \nabla) \mathbf{u} \approx (\mathbf{u}_{i-1} \cdot \nabla) \mathbf{u}_i$  until convergence
- . A derived “pressure” equation is solved, starting from the discretized momentum equation and substituting it in the continuity equation – here one or more (pressure) correction may be needed

PIMPLE is used for unsteady (transient) calculation **for better accuracy** when nonlinearity of the convective term is strong



In fvSolution, you can specify the options for PIMPLE

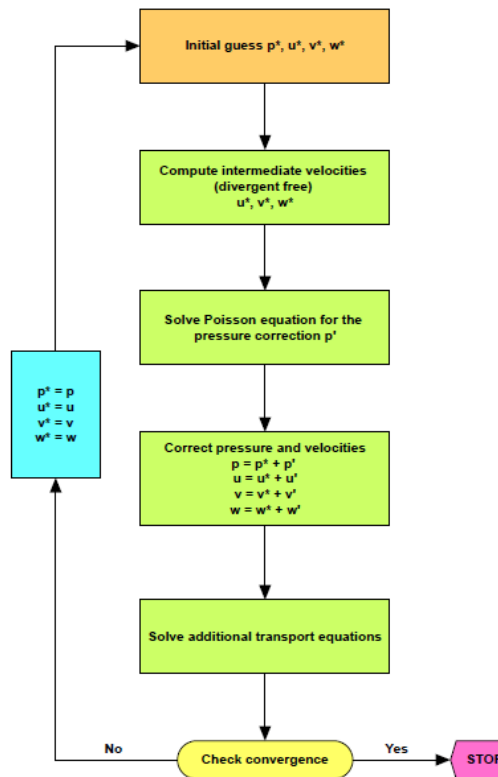
. # of nonOrthogonalCorrection  
(for mesh non-orthogonality)

. # of (pressure) nCorrectors  
Usually 2 (the first pressure corrector will  
create a conservative velocity field,  
while the second and following will establish  
the pressure distribution)

```
PIMPLE
{
    momentumPredictor    yes;
    nOuterCorrectors     1;
    nCorrectors           2;
    nNonOrthogonalCorrectors 1;
}
```

. # nOuterCorrectors  
Help the convergence of the  
nonlinear terms especially  
when using large time step

## SIMPLE is used for steady-state calculation



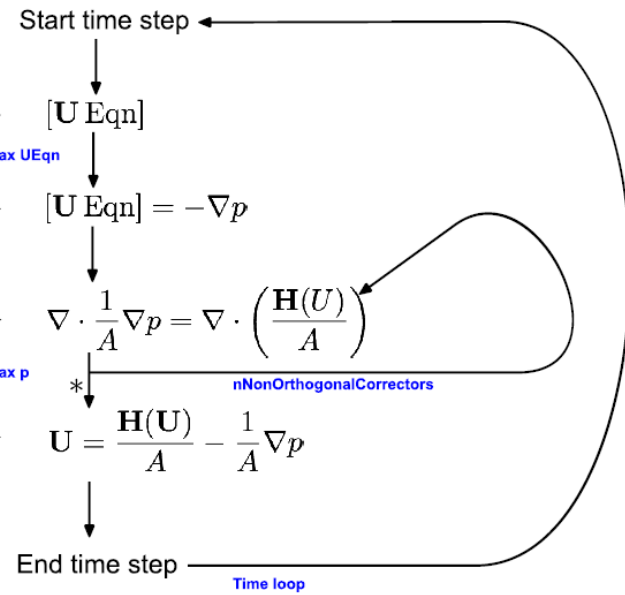
```

fvVectorMatrix UEqn
(
    fvm::ddt(U) + fvm::div(phi, U) - fvm::laplacian(nu, U)
);

solve(UEqn == -fvc::grad(p));

fvScalarMatrix pEqn
(
    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
);

U = HbyA - rAU*fvc::grad(p);
  
```



$$* \phi = \mathbf{S}_f \cdot [(\mathbf{H}/A)_f - (1/A)_f(\nabla p)_f]$$



# OpenFOAM

In fvSolution, you can specify the options for SIMPLE

- . # of nonOrthogonalCorrection
- . One (pressure) correction
- . Relaxation factors
- . Residual controls

$$\phi_P^n = \phi_P^{n-1} + \alpha(\phi_P^{n*} - \phi_P^{n-1})$$

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

SIMPLE

```
{
  nNonOrthogonalCorrectors 1;
}
```

SIMPLE

```
{
  nNonOrthogonalCorrectors 2;
  consistent yes;

  pRefCell      0;
  pRefValue     0;

  residualControl
  {
    p          1e-7;
    "(U|k|epsilon)" 1e-7;
  }
}
```

# First steps with OF

I am curious about OpenFOAM ... but which version?

Open  FOAM®

[openfoam.com](http://openfoam.com)

This is the version we will  
use (OpenFoam-v2306)

 OpenFOAM

[openfoam.org](http://openfoam.org)

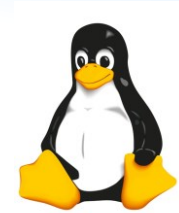
## IMPORTANT!


If you want to use an available solver, or take features from available solvers for your own solver, be very careful and select the right OF version!

# First steps with OF

## Can I use it on my computer?

OpenFOAM runs natively on Linux systems...





**Ubuntu**  
[Canonical Group Limited](#)

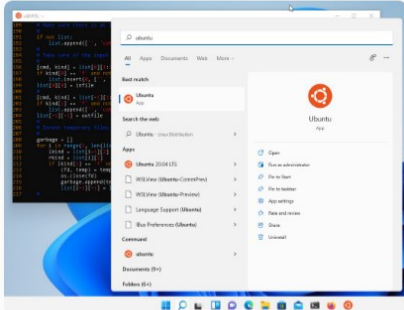
Ottieni

5,0 ★  
Media

1  
Classificazioni

Install a complete Ubuntu terminal environment

Screenshot



Descrizione

Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL), infrastructure without leaving Windows.

Key features:

- Efficient command line utilities including bash, ssh, git, apt, npm, pip and many more
- Manage Docker containers with improved performance and startup times

MAC, or the Linux subsystem for Windows can be used

# First steps with OF



## How to get OpenFOAM?

Follow the simple steps on the download page

(example for OF-9 from the .org version)

### Installation

OpenFOAM and *ParaView* can be simply installed for the first time using the **apt** package management tool. The user will need to provide superuser password authentication when executing the following commands with **sudo**

1. **Copy and paste** the following in a **terminal prompt** (*Applications* → *Accessories* → *Terminal*) to add **d1.openfoam.org** to the list of software repositories for **apt** to search, and to add the public key (**gpg.key**) for the repository to enable package signatures to be verified.

**Note:** use secure **https://** for the public key to ensure secure transfer, but use **http://** for the repository, since **https://** may not be supported and is not required since the key provides secure authentication of the package files.

```
sudo sh -c "wget -O - https://d1.openfoam.org/gpg.key | apt-key add -"  
sudo add-apt-repository http://d1.openfoam.org/ubuntu
```

**\*\*Note:** This only needs to be done once for a given system

2. Update the **apt** package list to account for the new download repository location

```
sudo apt-get update
```

3. Install OpenFOAM (9 in the name refers to version 9) which also installs **paraviewopenfoam56** as a dependency.

```
sudo apt-get -y install openfoam9
```

OpenFOAM 9 and *ParaView* 5.6.3 are now installed in the `/opt` directory.

# First steps with OF

## How to get OpenFOAM?

Follow the simple steps on the download page

(example for v2306 from the .com version)

<https://develop.openfoam.com/Development/openfoam/-/wikis/precompiled/debian>

### debian



Precompiled packages (Debian, Ubuntu)


#### Quick-start:

```
# Add the repository
curl https://dl.openfoam.com/add-debian-repo.sh | sudo bash

# Update the repository information
sudo apt-get update

# Install preferred package. Eg,
sudo apt-get install openfoam2306-default

# Use the openfoam shell session. Eg,
openfoam2306
```

 The packages do **not** contain *visualization* (eg, ParaView/runTimePostProcessing) or *external-solver* (eg, PETSc) modules: see the [corresponding FAQ](#)

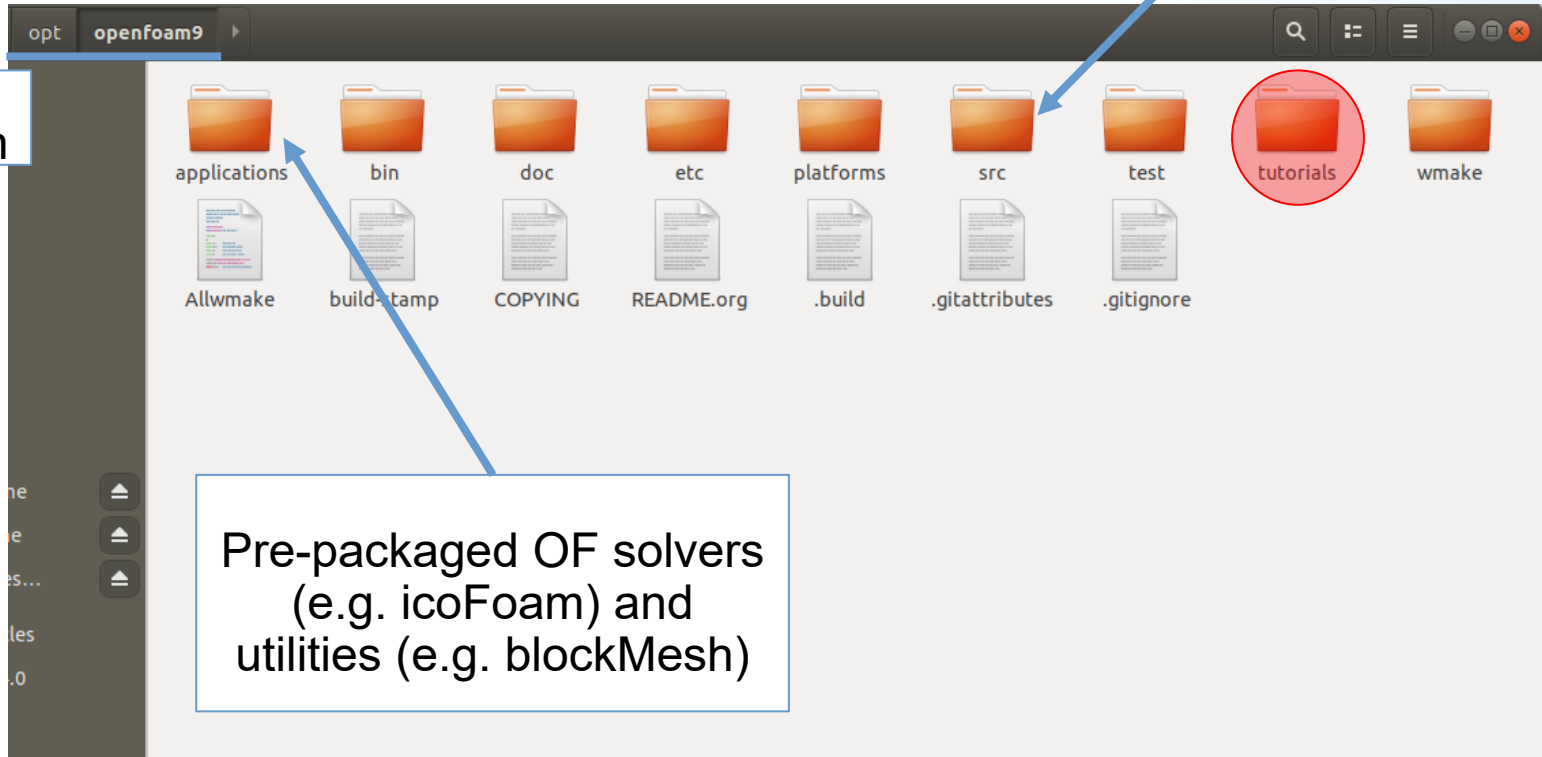
- Debian / Ubuntu
  - OpenFOAM repository
  - Supported versions and distributions
  - Sub-packages
  - Installation locations

# First steps with OF

## What comes with OpenFOAM?

Main OF library

Typical location



Pre-packaged OF solvers  
(e.g. icoFoam) and  
utilities (e.g. blockMesh)

# First steps with OF

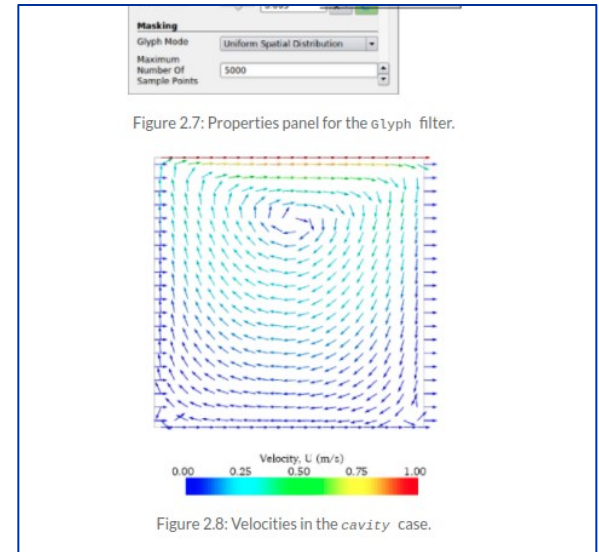
## Learn OpenFOAM - Official documentation

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



The screenshot shows the CFD Direct website. The header includes the logo and navigation links: Home, Book, OpenFOAM, Cloud. The main content area is titled "OpenFOAM v9 User Guide: 2 Tutorials". Below the title are links for "Table of Contents", "Index", and versions "Version 9", "Version 8", "Version 7", "Version 6", "Version 5", and "Version 4". There are also "prev" and "next" navigation links. The main heading is "Chapter 2 Tutorials". The text below describes the process of setup, simulation, and post-processing for OpenFOAM test cases. It mentions the `$FOAM_TUTORIALS` directory and the `$FOAM_RUN` environment variable. A code block shows the command `ls $FOAM_RUN`. A note at the bottom states: "If a message is returned saying no such directory exists, the user should create the directory by typing".

It includes some post-processing examples



# First steps with OF

## Learn OpenFOAM - Overview of Finite Volume Method from H. Jasak

[https://www.youtube.com/watch?v=a4B\\_oXR5Kzs&ab\\_channel=KennethHoste](https://www.youtube.com/watch?v=a4B_oXR5Kzs&ab_channel=KennethHoste)

### Diffusion Discretisation

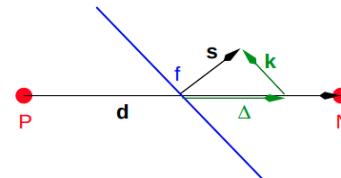
WIKI

Diffusion Operator and Mesh Non-Orthogonality

- Diffusion term is discretised using the Gauss Theorem

$$\oint_S \gamma(\mathbf{n} \cdot \nabla \phi) dS = \sum_f \int_{S_f} \gamma(\mathbf{n} \cdot \nabla \phi) dS = \sum_f \gamma_f \mathbf{s}_f \cdot (\nabla \phi)_f$$

- Evaluation of the face-normal gradient. If  $\mathbf{s}$  and  $\mathbf{d}_f = \overline{PN}$  are aligned, use difference across the face. For non-orthogonal meshes, a correction term may be necessary



$$\mathbf{s}_f \cdot (\nabla \phi)_f = |\mathbf{s}_f| \frac{\phi_N - \phi_P}{|\mathbf{d}_f|} + \mathbf{k}_f \cdot (\nabla \phi)_f$$



# First steps with OF

## Learn OpenFOAM – Presentations and tutorial from OF wiki

[https://wiki.openfoam.com/index.php?title=%223\\_weeks%22\\_series](https://wiki.openfoam.com/index.php?title=%223_weeks%22_series)

3-weeks-series

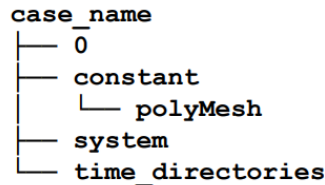
Day 1	Day 2	Day 3	Day 4	Day 5
<a href="#">install - first steps</a>	<a href="#">steps - visualization</a>	<a href="#">introductory course</a>	<a href="#">discretization</a>	<a href="#">theory - fun simulations - tips</a>
Day 6	Day 7	Day 8	Day 9	Day 10
<a href="#">geometry and meshing</a>	<a href="#">turbulence 1</a>	<a href="#">turbulence 2</a>	<a href="#">multiphase</a>	<a href="#">parallelization</a>
Day 11	Day 12	Day 13	Day 14	Day 15
<a href="#">programming 1</a>	<a href="#">programming 2</a>	<a href="#">programming 3</a>	<a href="#">programming 4</a>	<a href="#">programming 5</a>

# First steps with OF

## Learn OpenFOAM - Presentations from Wolf Dynamics

### Running my first OpenFOAM® case setup blindly

Before we start – Always remember the directory structure



- To keep everything in order, the case directory is often located in the path `$WM_PROJECT_USER_DIR/run`.
- This is not compulsory but highly advisable, you can put the case in any directory of your preference.
- The name of the case directory if given by the user (do not use white spaces).
- You run the applications and utilities in the top level of this directory.
- The directory **system** contains run-time control and solver numerics.
- The directory **constant** contains physical properties, turbulence modeling properties, advanced physics and so on.
- The directory **constant/polyMesh** contains the polyhedral mesh information.
- The directory **0** contains boundary conditions (BC) and initial conditions (IC).

### Solution initialization using codeStream

Body of the **codeStream** directive for initial conditions

```
internalField #codeStream
{
  codeInclude
  #{
    #include "fvCFD.H"
  };

  codeOptions
  #{
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude
  };

  codeLibs
  #{
    -lmeshTools \
    -lfiniteVolume
  };

  code
  #{
    #};
  };
};
```

Use **codeStream** to set the value of the initial conditions

Files needed for compilation

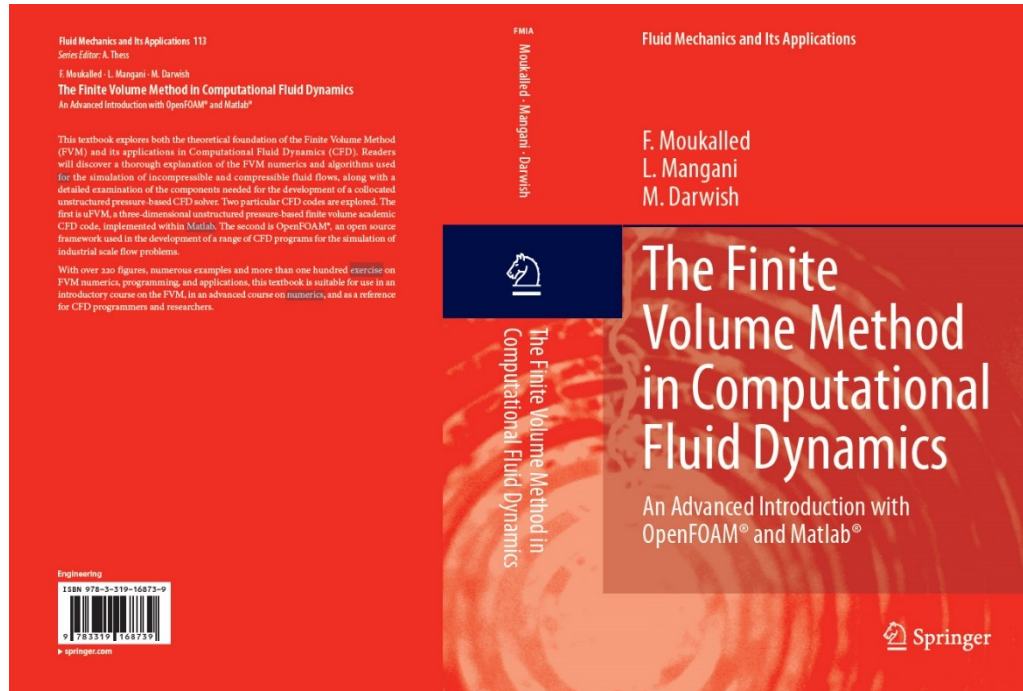
Compilation options

Libraries needed for compilation. Needed if you want to visualize the output of the initial conditions at time zero

Insert your code here. At this point, you need to know how to access internal mesh information

# First steps with OF

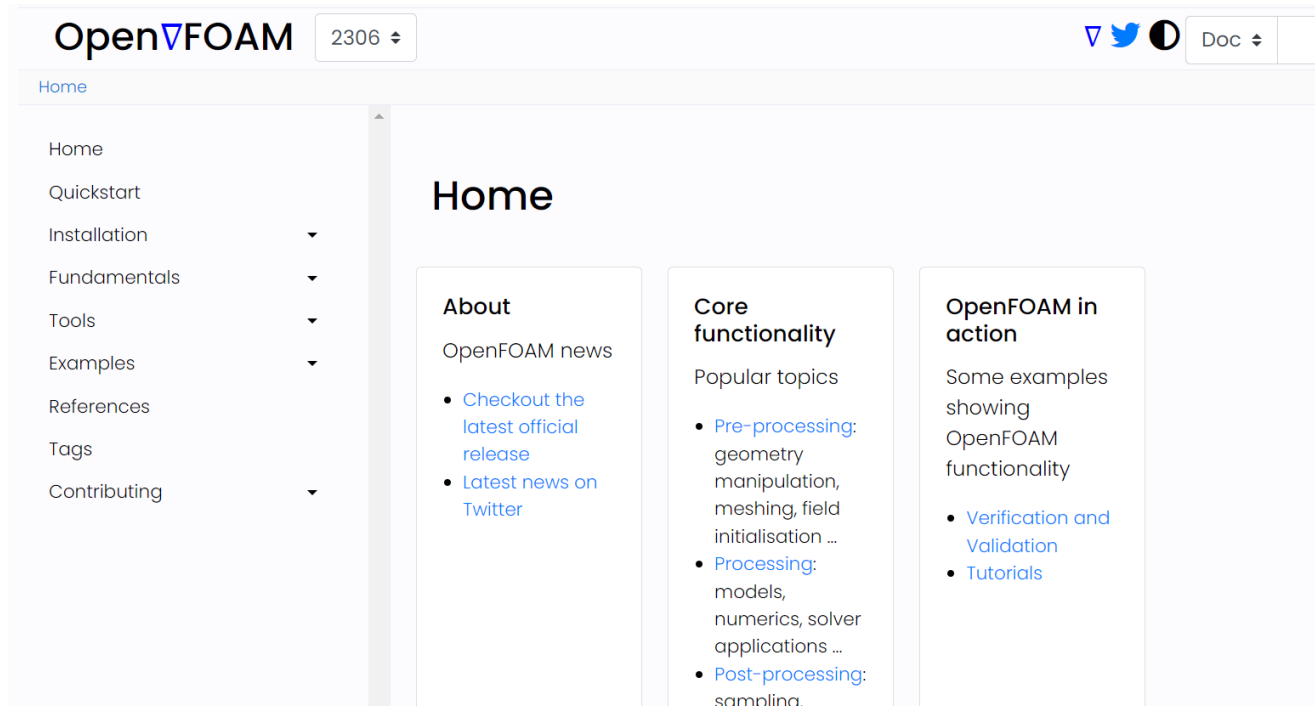
## Learn OpenFOAM – Book «F. Moukalled, L. Mangani, M. Darwish. The Finite Volume Method in Computational Fluid Dynamics. Springer, 2016»



# First steps with OF

## Learn OpenFOAM – Browse the C++ source guide official documentation

- <https://doc.openfoam.com/2306/>
- <https://cpp.openfoam.org/v9/>



The screenshot shows the OpenFOAM documentation website. The header includes the OpenFOAM logo, a version selector set to 2306, and social media icons for GitHub, Twitter, and YouTube. A navigation menu on the left lists: Home, Quickstart, Installation, Fundamentals, Tools, Examples, References, Tags, and Contributing. The main content area is titled 'Home' and contains three columns of information:

- About**  
OpenFOAM news
  - [Checkout the latest official release](#)
  - [Latest news on Twitter](#)
- Core functionality**  
Popular topics
  - [Pre-processing:](#) geometry manipulation, meshing, field initialisation ...
  - [Processing:](#) models, numerics, solver applications ...
  - [Post-processing:](#) sampling.
- OpenFOAM in action**  
Some examples showing OpenFOAM functionality
  - [Verification and Validation](#)
  - [Tutorials](#)

# First steps with OF

## Learn OpenFOAM – Plenty of additional resources

- Tutorials/lectures (have a look on Google or YouTube)
- Master/PhD thesis etc.
- Forums
- (Often) direct communication with solver developers

### And remember:

- **Don't get frustrated: there is always a way out with OpenFOAM and, most likely, someone who had your same problem and will be happy to help**
- **Don't get discouraged: the entry barrier may seem steep, but skills you'll learn will allow you to tackle any kind of problems**
- **If possible, do not do it alone!**

# Joint ICTP-IAEA Workshop on Open-Source Nuclear Codes for Reactor Analysis

7 August 2023 – 11 August 2023

*Thank you!*