INFORMATICS INSTITUTE

OPENFOAM®'A GİRİŞ ÇALIŞTAYI

Temel OpenFOAM Bilgileri

Emre Cenk Ersan

ITU Computational Biomechanics Research Group
https://valve.be.itu.edu.tr/

Workshop Supervisor: Prof. Dr. M. Serdar Çelebi

December 1, 2023

Overview

- We will look at the following in this section:

1. **OpenFOAM File Structure**
    i. **Installation directory structure**
    ii. **User’s working directory**
    iii. **Application directory structure**
2. **Solvers**
3. **Boundary and initial conditions**
4. **Numerical schemes**
5. **Linear solvers**
6. **Utilities**

Installation Directory Structure

- **Before start working, we need to source the OpenFOAM software version**
  - Connect your system with the OpenFOAM installation path in each new terminal session
  - OpenFOAM system variables and aliases are loaded by typically typing e.g:

```
# source installPath/openfoam/openfoamversion/etc/bashrc
```

  - For OpenFOAM ESI v2206 specifically (with default installation path)

```
# source /usr/lib/openfoam/openfoam2206/etc/bashrc
```

  - Optionally, you can add this line to your `/home/user/.bashrc` with a new alias and call it when needed:

```
alias of2206=‘source /usr/lib/openfoam/openfoam2206/etc/bashrc’
 # of2206
```

Installation Directory Structure

$WM_PROJECT_DIR

∟ ***applications***	Applications and their sources
∟ ***solvers***	Source codes of solvers
∟ ***test***	Sample codes
∟ ***utilities***	Source codes of utilities
∟ ***bin***	Scripts for compiler
∟ ***build***	MPI library configuration
∟ ***etc***	Configuration files
∟ ***META-INFO***	Build information
∟ ***platforms***	Compiled libraries and binaries
∟ ***src***	Source codes for libraries
∟ ***tutorials***	Tutorials
∟ ***wmake***	Directives for compiler

- **System variables and aliases**

```
alias foam='cd $WM_PROJECT_DIR'
alias app='cd $FOAM_APP'
alias sol='cd $FOAM_SOLVERS'
alias util='cd $FOAM_UTILITIES'
alias lib='cd $FOAM_LIBBIN'
alias src='cd $FOAM_SRC'
alias tut='cd $FOAM_TUTORIALS'
```

- For all predefined aliases set by OpenFOAM, see

```
$WM_PROJECT_DIR/etc/config.sh/aliases
```

User’s working directory

- Each OpenFOAM user has his/her own working directory.
- The path to user working directory:

```
WM_PROJECT_USER_DIR=/home/user/OpenFOAM/{user}-dev
```

- You can test the location by typing in terminal:

```
echo $WM_PROJECT_USER_DIR
```

- User directory is usually being used for code modifications and developing own applications.
- It is useful to create same structure in the user directory as it is in the installation directory.

${user}-dev

```
${user}-dev
   ∟ applications        →  User application source codes
      ∟ solvers
      ∟ utilities
   ∟ platforms
      ∟ platformName
         ∟ bin           →  User application binaries
         ∟ lib           →  User libraries
   ∟ run                 →  User projects
```

```
alias ufoam='cd $WM_PROJECT_USER_DIR'
```

Other useful aliases:

```
uapp, usol, uutil, run
```

Application directory structure

$WM_PROJECT_DIR/applications/solvers

∟ ***appName***	
∟ *appName.C*	Source code of the solver
∟ *createFields.H*	Field variable declerations
∟ ***Make***	Compilation instructions
∟ *options*	Specifies libraries and their locations
∟ *files*	Names all source files and specifices the application name.

$WM_PROJECT_DIR/applications/utilities

∟ ***appName***
∟ *appName.C*
∟ *header_files.H*
∟ ***Make***
∟ *options*
∟ *files*

- To create your own applications, it is recommended to create a copy of the application directory structure in user working directory and modify ***Make****/files* and ***Make****/options*.

Solvers

./applications/solvers

- ⌞ **acoustic**
- ⌞ ***basic***
- ⌞ ***combustion***
- ⌞ ***compressible***
- ⌞ ***discreteMethods***
- ⌞ ***DNS***
- ⌞ ***electromagnetics***
- ⌞ ***financial***
- ⌞ ***finiteArea***
- ⌞ ***heatTransfer***
- ⌞ ***incompressible***
- ⌞ ***lagrangian***
- ⌞ ***multiphase***
- ⌞ ***stressAnalysis***

➢ Basic CFD solvers:

laplacianFoam	Solves a Laplace equation
potentialFoam	Solves a potential flow.
scalarTransportFoam	Solves a transport equation for a passive scalar.

➢ Incompressible flow solvers:

icoFoam	Transient solver for incompressible, laminar flow of Newtonian fluids.
simpleFoam	Steady state solver for incompressible, turbulent flow.
pisoFoam	Transient solver for incompressible, turbulent flow.
pimpleFoam	Transient solver for incompressible, turbulent flow.
nonNewtonianIcoFoam	Transient solver for incompressible, laminar flow of non-Newtonian fluids

Solvers

- Compressible flow solvers:

rhoCentralFoam	Density based compressible flow solver based on central upwind scheme.
sonicFoam	Transient solver for trans-sonic/supersonic, laminar/turbulent flow of a compressible gas.

- Multiphase flow solvers:

interFoam	Solver for two incompressible, isothermal and immiscible fluids based on the VOF (Volume of Fluid) method.
bubbleFoam	Solver for a system of two incompressible fluid phases with one phase dispersed, e.g. gas bubbles in a liquid
multiPhaseEulerFoam	Solver for multiple compressible miscible fluid phases with heat transfer.

- Other solvers:

solidDisplacementFoam	Transient solver for linear-elastic, small-strain solver deformation of a solid body.
mdFoam	Molecular dynamics solver for fluids.
buoyantSimpleFoam	Steady state solver for buoyant, turbulent flow for compressible fluids.

Boundary and initial conditions

- Setting appropriate boundary conditions is vital for a successful simulation.
- Ill-posed boundary conditions will lead to physically incorrect predictions and solver failure.
- For each solved field, users must specify the boundary conditions.
- OpenFOAM distinguish between base type and primitive boundary conditions.

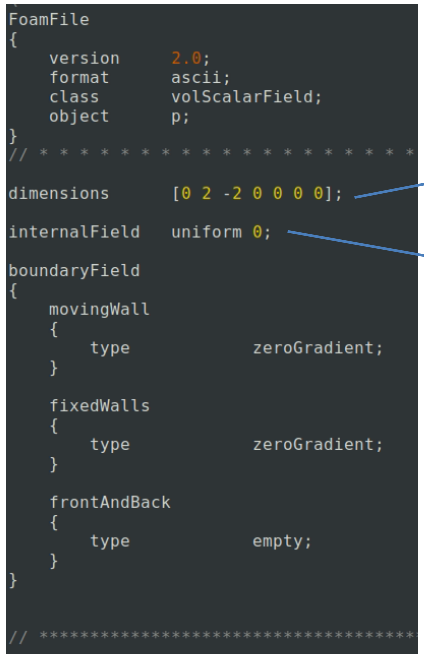
Base type:	Based on geometry information
Primitive type:	Assigns the value of the field variables in the given patch.

- Base type boundary conditions are defined in the file ***boundary*** in ***constant/polyMesh*** directory.
- Primitive type boundary conditions which are Dirichlet, Neumann and Robin boundary conditions, are defined in the field variables dictionaries located in the directory **0**.
- Base type boundary condition which are constrained or paired (same as in *boundary* file and in field variable dictionaries): **symmetry, symmetryPlane, empty, wedge, cyclic, processor**
- Base type **patch** can be any of the primitive of derived type boundary conditions:

  fixedValue, zeroGradient, inletOutlet, slip, totalPressure, supersonicFreeStream …
- Base type **wall** is defined as: ***noSlip*** in **U**, **zeroGradient** in **p**.

Boundary and initial conditions

- A typical initial & boundary condition file for **pressure**:



```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * *

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************
```

Parameter ***dimensions*** stands for physical dimensions according to SI unit system.

No	Property	SI unit
1	Mass	Kilogram (kg)
2	Length	Meter (m)
3	Time	Second (s)
4	Temperature	Kelvin (K)
5	Quantity	Kilogram-mole (kgmol)
6	Current	Ampere (A)
7	Lum. intensity	Candela (cd)

Parameter ***internalField*** defines values in cell volumes of the mesh.

Values can be constant or nonuniform with a list of values.

- Typical boundary conditions are as follows,

  Inlet: p – zeroGradient, U – fixed value, **Outlet:** p – fixedValue, U – zeroGradient, **Wall:** p – zeroGradient, U - noSlip

- For a detailed list of boundary conditions, please visit:

  https://www.openfoam.com/documentation/user-guide/a-reference/a.4-standard-boundary-conditions

Numerical schemes

- Classical incompressible Navier-Stokes equation:

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) - \mu \nabla^2 \boldsymbol{U} = -\nabla p + \boldsymbol{b}$$

- Navier-Stokes equation in OpenFOAM:

```
solve
(
        fvm::ddt(rho,U)
        + fvm::div(phi,U)
        - fvm::laplacian(mu,U)
        ==
        - fvc::grad(p)
        + b
)
```

- OpenFOAM convers the PDEs into a set of linear algebraic equations, [A]{x} = {b}.
- {x} and {b} are volFields and [A] is a fvMatrix, created by the disctretization of a geometric field and inherits the algebra of it.
- solve() performs inversion to solve [A]{x} = {b} for one step.
- U is a volVectorField defined on a mesh.
- fvm (Finite Volume Method) returns a matrix.
- fvc (Finite volume Calculus) returns a geometricField.
- ddt( ) -> time derivative ($\frac{\partial \emptyset}{\partial t}$)
- div( ) -> divergence ($\nabla \cdot (\emptyset \boldsymbol{U})$).
- laplacian( ) -> Laplacian ($\mu \nabla^2 \boldsymbol{U}$).
- grad( ) -> gradient (∇p).

Numerical schemes

- Discretization schemes are defined in ***fvSchemes*** file.

- **Time discretization schemes (ddtSchemes):**

Euler	time dependent, first order, implicit, bounded.
CrankNicolson	time dependent, second order, implicit, bounded.
backward	time dependent second order, implicit, might be unbounded.
steadyState	steady state.

- **Interpolation schemes (interpolationSchemes):**

linear	linear interpolation, central differencing.
upwind	upwind difference.
linearUpwind	Linear upwind differencing.
limitedLinear	limited linear differencing.

Numerical schemes

- **Convective term discretization schemes (divSchemes):**

Gauss <interpolationScheme>

upwind	first order accurate, bounded.
linearUpwind	First/second order accurate, bounded.
linear	second order accurate, unbounded.

- **Gradient term discretization schemes (gradSchemes):**

Gauss <interpolationScheme>	second order accurate Gaussian integration
leastSquares	second order accurate least squares.

- **Surface normal gradient schemes (snGradSchemes)**

orthogonal	for perfect hexahedral meshes, no non-orthogonal correction.
corrected	non-orthogonal correction.
uncorrected	no non-orthogonal correction.
limited	limited non-orthogonal correction.

Numerical schemes

➢ **Laplacian term discretization schemes (laplacianSchemes):**

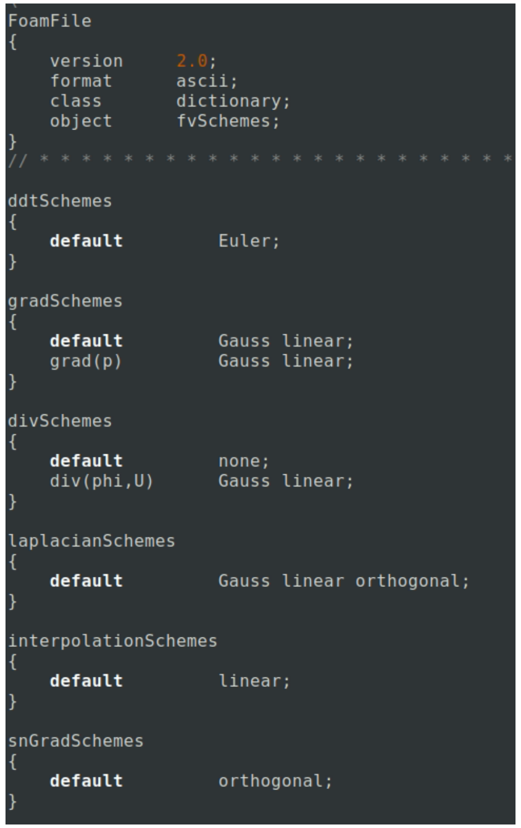
Gauss <interpolationScheme> <snGradScheme>

corrected	second order, unbounded, conservative
uncorrected	first order, bounded, non-conservative
limited	blend of correct and uncorrected.

- First order methods are bounded and stable but diffusive.
- Second order methods are accurate but might become oscillatory.
- We want a **second order accurate solution**.
- We look for **Accuracy, Stability** and **Boundedness**.

➢ For more details about numerical schemes, please visit

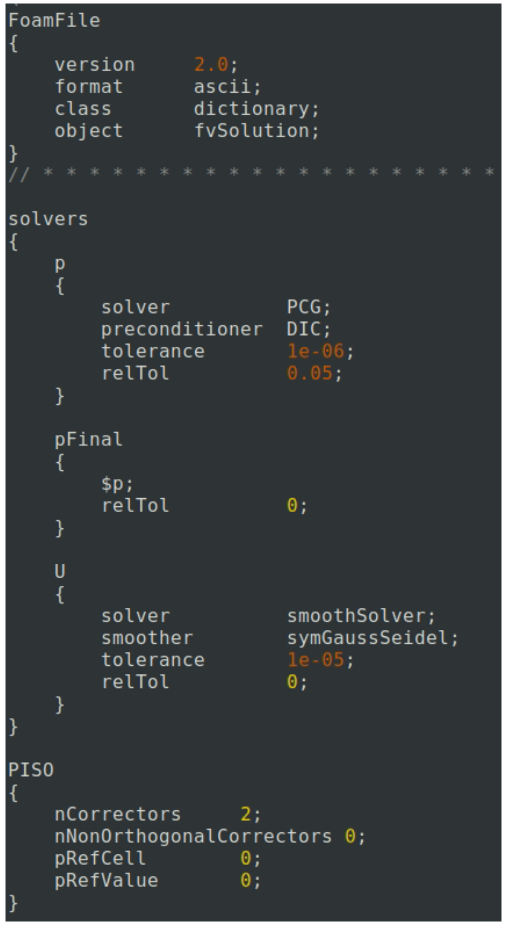
https://www.openfoam.com/documentation/user-guide/6-solving/6.2-numerical-schemes



```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * *

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear orthogonal;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         orthogonal;
}
```

Linear solvers

- After spatial and temporal discretization in every control volume of the domain, a system of linear equations [A]{x} = {b} for the transported quantity Ø can be solved.
- The system can be solved by using any iterative or direct method.
- The equation solvers, tolerances and algorithms are controlled from the sub-dictionary solvers located in ***fvSolution*** dictionary file.
- In this generic case, we PCG method with the DIC preconditioner to solve for pressure for an absolute tolerance of 1e-06 and a relative tolerance of 0.05.
- The entry pFinal refers to the final pressure correction.
- To solve the velocity field U, we use smoothsolver with smoother symGaussSeidel.
- We need to know that solving for velocity is relatively inexpensive compared to solving for pressure.
- The pressure equation governs the mass conservation.
- Selection of the tolerance according to the complexity of the problem has paramount importance.



```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * *

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.05;
    }

    pFinal
    {
        $p;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

Linear solvers

- The symmetry of the [A] matrix depends on the structure of the equation.
- Pressure is a symmetric matrix and velocity is an asymmetric matrix.
- The linear solvers are iterative, they are based on reducing the residual over a succession of solutions.
- After each solver iteration the residual is reevaluated.
- The solver stops if

    the residual drops below the absolute tolerance,

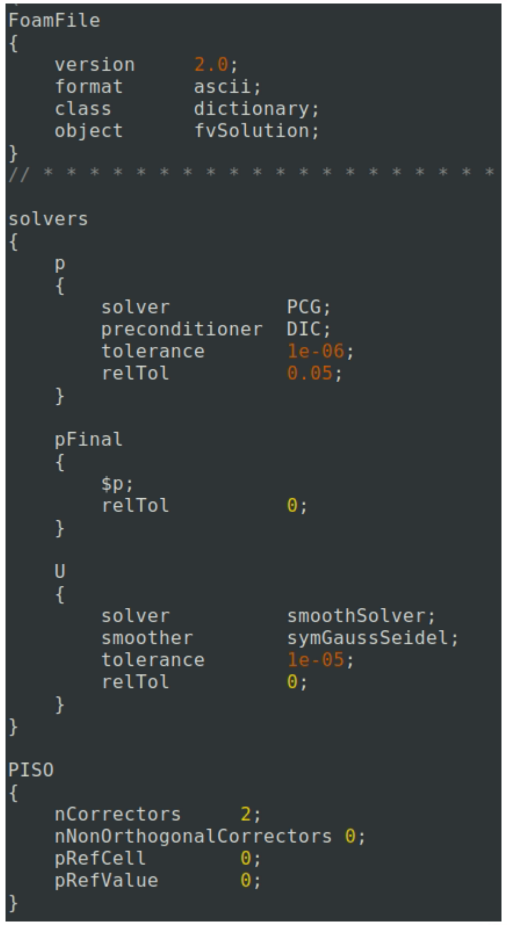
    the ratio of initial residuals falls below the relative tolerance,

    the number of iterations exceeds a maximum number of iterations (default value is 1000).
- Linear solvers:

    PCG, PBiCG, GAMG, smoothSolver, …
- Preconditioners:

    diagonal, DIC, DILU, FDIC, GAMG, …
- Smoothers:

    DIC, DICGaussSeidel, GaussSeidel, …



```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * *

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.05;
    }

    pFinal
    {
        $p;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

Utilities

- The utilities with the OpenFOAM distribution are in the `$FOAM_UTILITIES` directory and can be reached by `util` command.

./applications/utilities

- ⌊ ***mesh***
- ⌊ **miscellaneous**
- ⌊ **parallelProcessing**
- ⌊ **postProcessing**
- ⌊ **preprocessing**
- ⌊ **surface**
- ⌊ **thermophysical**

- **Pre-processing**

mapFields	Maps volume fields from one mesh to another.
setFields	Set values on a selected set of cells/patch faces.

- **Meshing**

blockMesh	
checkMesh	
fluentMeshToFoam	Converts a Fluent mesh to OpenFOAM format
foamMeshToFluent	Writes out the OpenFOAM mesh in Fluent mesh format
transformPoints	Translate, rotate or scale mesh.

- **Parallel processing**

decomposePar	Decomposes a mesh and fields for parallel execution
reconstructPar	Reconstructs a decomposed mesh and fields of a case.

- For more details about standard utilities, please visit

  https://www.openfoam.com/documentation/user-guide/a-reference/a.2-standard-utilities

Thank you ☺

Emre Cenk ERSAN
PhD Candidate – Research Assistant
Istanbul Technical University
Computational Science and Engineering
ersane@itu.edu.tr