XinFOAM

December 17, 2019

1 Numerical

OpenFOAM applications are designed for use with unstructured meshes, offering up to second oder accuracy, predominately using collocated variable arrangements. Most focus on the Finite Volume Method, for which the conservative form of the general scalar transport equation for the property ϕ takes the form:

$$\underbrace{\frac{\partial}{\partial t}(\rho\phi)}_{\text{unsteady}} + \underbrace{\nabla\cdot(\rho\phi\mathbf{u})}_{\text{convection}} = \underbrace{\nabla\cdot(\Gamma\nabla\phi)}_{\text{diffusion}} + \underbrace{S_\phi}_{\text{source}} \tag{1}$$

unsteady:	time dependent
convection	velocity dependent
diffusion	viscosity dependent
source	heat, force...

The finite volume method requires the integration over a 3D-control volume, such that:

$$\int_V \frac{\partial}{\partial t}(\rho\phi)dV + \int_V \nabla\cdot(\rho\phi\mathbf{u})\,dV = \int_V \nabla\cdot(\Gamma\nabla\phi)\,dV + \int_V S_\phi dV \tag{2}$$

or in a more compact way:

$$\boldsymbol{A}\mathbf{x} = \mathbf{b} \tag{3}$$

where

A:	coefficient matrix
x	vector of unknows dependent
b	source vector

The discretisation process employs user selected schemes to build the A matrix and b vector, described in the following sections. Choice of schemes are set in the *fvSchemes* dictionary.

OpenFOAM includes a wide range of solution and scheme controls, specified via dictionary files in the case system sub-directory. These are described by:

Numerical schemes: The treatment of each term in the system of equations is specified in the fvSchemes dictionary. This enables fine-grain control of e.g. temporal, gradient, divergence and interpolation schemes.

Linear equation solvers Solution methods: Case solution parameters are specified in the fvSolution dictionary. These include choice of linear equation solver per field variable, algorithm controls e.g. number of inner and outer iterations and under-relaxation.

Finite volume options: Additional run-time selectable physical modelling and general finite terms are prescribed in the fvOptions dictionary, targeting e.g. acoustics, heat transfer, momentum sources, multi-region coupling, linearised sources/sinks and many more.

1.1 Schemes

1.2 Temporal schemes

OpenFOAM includes a variety of schemes to integrate fields with respect to time:

Time schemes

Time schemes define how a property is integrated as a function of time, i.e.

$$\frac{\partial}{\partial t}(\phi) \tag{4}$$

Depending on the choice of schemes, field values at previous time steps are required, represented in the following as ϕ^0 and ϕ^{00} for the old and older time levels.

Usage

Time schemes properties are input in the *fvSchemes* file under the *ddtSchemes* sub-directory using the syntax:

```
ddtSchemes
{
    default         none;
    ddt(Q)          <time scheme>;
}
```

Available time schemes include

- Backward time scheme

$$\frac{\partial}{\partial t}(\phi) = \frac{1}{\Delta t}(\frac{3}{1}\phi - 2\phi^0 + \frac{1}{2}\phi^{00}) \tag{5}$$

  Using implicit, second order accuracy, transient, boundedness not guaranteed, conditionally stable.

  Usage

```
ddtSchemes
{
    default         backward;
    ddt(phi)        backward;
}
```

- Crank-Nicolson time scheme Second order, Transient, Bounded, when using uniform time steps the schemes equate to:

$$\frac{\partial}{\partial t} = \frac{\phi - \phi^{00}}{2\Delta t} \tag{6}$$

  Usage

  The schemes is specified using:

```
ddtSchemes
{
    default         CrankNicolson <coeff>
    ddt(phi)        CrankNicolson <coeff>;
}
```

  The coefficient provides a blending between Euler and Crank-Nicolson schemes:

  0: Euler 1: Crank-Nicolson A value of 0.9 is a good compromise between accuracy and robustness

- Euler implicit time scheme

  Implicit, First order, Transient

$$\frac{\partial}{\partial t} = \frac{\phi - \phi^0}{\Delta t} \tag{7}$$

  Usage

```
ddtSchemes
{
    default         Euler;
    ddt(phi)        Euler;
}
```

- Local Euler implicit/explicit time scheme First order Pseudo transient, designed for steady cases Spatially varying, cell-based time scale set by specific Local Time Stepping (LTS) solvers

  Usage

```
ddtSchemes
{
    default         localEuler;
    ddt(phi)        localEuler;
}
```

- Steady state time scheme Steady state, set temporal derivative contributions to zero,

$$\frac{\partial}{\partial t} = 0 \tag{8}$$

1.3 Spatial schemes

At their core, spatial schemes rely heavily on *interpolation schemes* to transform cell-bases qualities to cell faces, in combination with **Gauss Theorem** to convert volume integrals to surface integrals.

- Gradient Schemes
- Divergence schemes
- Laplacian schemes
- Surface-normal gradient schemes

1.4 Wall distance calculation method

Distance to the nearest wall is required, e.g. for a number of turbulence models. Several calculation methods are available:

- Mesh-wave wall distance
- Poisson wall distance

1.5 Solutions

After using different schemes to build the equation systems. The next step is to chose a technic to solve the linear equations, the matrix sytem.

$$x = A^{-1}b \tag{9}$$

where the inverse of the diagonal matrix is simply:

$$A^{-1} = \frac{1}{dia(A)} \tag{10}$$

This is available as the diagonalSolver. More tyically the matrix cannot be inverted easily and the system is solved using iterative methods, as discribed in the following sections.

1.6 Solvers

- Smooth solvers
- Conjugate gradient solvers
- Multigrid solvers

1.7 Solver control

- Under relaxation
- Residuals
- Case termination

Common usage minIter: minimum number of solver iterations maxIter: maximum number of solver iterations nSweeps: number of solver iterations between checks for solver convergence Implementation details Matrix structure Matrix coefficients are stored in upper-triangular order

neighbour cell index always higher than owner cell index across a face when looping over cell faces, the face index increases with increasing cell index for the 1-D case, if cell index 0 is at the boundary, this equates to a monotonic increase in cell numbers, i.e. defines a continuous sweep across the 1-D region used in Gauss-Seidel method

2 boundary condition

Boundary conditions are extremely important. Indeed, they cause the most common errors. OpenFOAM uses similar set-up as other solvers.

- At the inlet (flow inlet) to the computational domain total pressure and total temperature are set. The rest of variables are reconstructed.
- At the outlet the static pressure is set.
- At the rigid walls velocity is set to zero.
- Multiple Reference Frame (MRF) is used for rotational components.
- For steady-state computations the initial conditions have no influence to the results.
- For steady-state computations the initial conditions just help to make the case run.

The main variables to be set are following:

p: static pressure p

U: velocity vector U

T: static temperature

k: turbulent kinetic energy k

ω: specific turbulence dissipation

ϵ specific turbulence dissipation

The initial and boundary conditions for all variables are set in files located in directories named by numbers. Typically directory 0 is recommended to start a simulation from.

Initial conditions are set in parameter internalField putting the values into the cell centres.

At boundaries, initial conditions are set individually by parameter value.

Following table shows recommended model of boundary conditions for computed variables:

boundaty type	p	U	T	k	ω / ϵ
STATOR:					
inlet	totalPressure	pressureDirected InletVelocity	totalTemperature	turbulentIntensity KineticEnergyInlet	fixedValue
volute_wall	zeroGradient	fixexValue	compressible:: turbulentTemperature CoupledBaffeMixed	compressible:: kqRWallFunction	compressible:: omegaWallFunction
ring*	mixingPlane	inletOutlet	inletOutlet	inletOutlet	inletOutlet
ROTOR					
outlet	fixedMeanValue	inletOutlet	inletOutlet	inletOutlet	inletOutlet
outlet wall	zeroGradient	fixedValue	compressible:: turbulentTemperature CoupledBaffeMixed	compressible:: kqRWallFunction	compressible:: omegaWallFunction
wheel wall	zeroGradient	fixexValue	zeroGradient	compressible:: kqRWallFunction	compressible:: omegaWallFunction
ring*	zeroGradient	mixingPlaneVelocity	mixingPlane	mixingPlane	mixingPlane
SOLID:					
rotor inner wall	x	x	compressible:: turbulentTemperature CoupledBaffeMixed	x	x
stator inner wall	x	x	compressible:: turbulentTemperature CoupledBaffeMixed	x	x
outer wall	x	x	zeroGradient	x	x

Table 1: Boundary condition recommondation*

*: taken from: https://www.cfdsupport.com/Turbomachinery-CFD-manual/node283.html#13201

The shortcuts from the above table have following meaning:

- tP - totalPressure constant, e.g. 250 000 Pa, gamma = 1.2853 [-] is specific heat ratio
- pDIV - pressureDirectedInletVelocity, velocity is computed from difference between total and static pressure, inletDirection is velocity vector to be specified
- tT - totalTemperature, constant, e.g. 1050 K, gamma= 1.2853 [-] is specific heat ratio
- tIKEI - turbulentIntensityKineticEnergyInlet, intensity = 0.02, corresponds to turbulence intensity 2
- fV - fixedValue, e.g. velocity at the wall (0 0 0), or omega at inlet
- fMV - fixedMeanValue, is the same as fixed value, e.g. for pressure, but certain freedom is allowed to keep the variable average equal to meanValue
- zG - zeroGradient, the flux of the variable is zero in direction perpendicular to the surface
- TWF - compressible::turbulentTemperatureCoupledBaffleMixed special boundary condition for temperature enabling heat transfer from other regions
- kWF - compressible::kqRWallFunction is a standard wall function for k for compressible flow
- oWF - compressible::omegaWallFunction is a standard wall function for omega for compressible flow
- mPV - mixingPlaneVelocity, averaged velocity is mapped from neighbour patch
- mP - mixingPlane, averaged variable is mapped from neighbour patch
- iO - inletOutlet is by default zeroGradient, but changes to fixedValue when velocity vector direction points inside the computational domain (backward flow)

Another recommended boundary conditions for each vairable (depending on the selected turbulence model) at each boundary type. Note the variations for Wall BCs for a High Reynolds model(y+ 30-100), and a Low Reynolds model(y+ 1).

	inlet	outlet	farfield	stationary walls	moving walls	rotating walls
p	**zeroGradient**	**fixedValue 0**	**outletInlet 0**	**zeroGradient**	**zeroGradient**	**zeroGradient**
U	**surfaceNormalFV** negative value for entry velocity	**zeroGradient** **inletOutlet**	**inletOutlet** free stream velocity	**fixedValue 0**	**fixedValue** velocity of the wall	**rotatingWallVelocity** rotational velocity of the wall
k	**fixedValue** k = $(3/2) * (UI)^2$ U:free stream velocity I: Turbulent intensity	**zeroGradient** **inletOutlet**	**inletOutlet** k = $(3/2) * (UI)^2$ U:free stream velocity I: Turbulent intensity	**kqRWallFunction** (for y+ 30 to 100) **fixedValue 0** (for y+ 1)	**kqRWallFunction** (for y+ 30 to 100) **fixedValue 0** (for y+ 1)	**kqRWallFunction** (for y+ 30 to 100) **fixedValue 0** (for y+ 1)
epsilon	**fixedValue** $\epsilon = 0.09 * k * \omega$	**zeroGradient** **inletOutlet**	**inletOutlet** $\epsilon = 0.09 * k * \omega$	**epsilonWallFunction** (for y+ 30 to 100) fixedValue 10^{-12} (for y+ 1)	**epsilonWallFunction** (for y+ 30 to 100) fixedValue 10^{-12} (for y+ 1)	**epsilonWallFunction** (for y+ 30 to 100) fixedValue 10^{-12} (for y+ 1)
omega	**fixedValue** $\omega = \rho * \frac{k}{\mu} * (\frac{\mu_t}{\mu})^{-1}$ ρ:Density k:Turbulence kinetic energy μ: viscosity μ_t:turbulent viscosity	**zeroGradient** **inletOutlet**	**inletOutlet** $\omega = \rho * \frac{k}{\mu} * (\frac{\mu_t}{\mu})^{-1}$ ρ:Density k:Turbulence kinetic energy μ: viscosity μ_t:turbulent viscosity	**omegaWallFunction**	**omegaWallFunction**	**omegaWallFunction**
nutTilda	**fixedValue** nuTilda = $\sqrt{3/2} * U * I * L$ U:free stream velocity I:turbulent intensity L:length scale	**zeroGradient** **inletOutlet**	**inletOutlet** nuTilda = $\sqrt{3/2} * U * I * L$ U:free stream velocity I:turbulent intensity L:length scale	**zeroGradient** (for y+ 30 to 100) **fixexValue 0** (for y+ 1)	**zeroGradient** (for y+ 30 to 100) **fixexValue 0** (for y+ 1)	**zeroGradient** (for y+ 30 to 100) **fixexValue 0** (for y+ 1)
nut	**calculated**	**calculated**	**calculated**	**nutkWallFunction** (for y+ 30 to 100) **nutUSpalding** **WallFunction** (for y+ 1)	**nutkWallFunction** (for y+ 30 to 100) **nutUSpalding** **WallFunction** (for y+ 1)	**nutkWallFunction** (for y+ 30 to 100) **nutUSpalding** **WallFunction** (for y+ 1)

Table 2: Boundary condition recommondation*

*: taken from: ANSA for CFD Brief User's Guide. v19.1.2

solver	transient	compressible	turbulence	heat transfer	buoyancy	combustion	multi phase	particles	dynamicMesh	multi region	fvOptions
boundaryFoam											
buoyantPimpleFoam	√	√	√	√	√						√
buoyantSImpleFoam			√	√	√	√					√
chemFoam	√			√		√					
chtMultiRegionFoam	√	√	√	√	√					√	√
coldEngineFoam	√	√	√	√		√			√		√
engineFoam	√	√	√	√		√			√		√
fireFoam	√	√	√	√	√	√				√	√
icoFoam	√										
interFoam	√		√				√		√		√
laplacianFoam	√										√
pimpleFoam	√		√						√		√
pisoFoam	√		√								√
potensialFoam											
reactingFoam	√	√	√	√		√					√
reactingParcelFoam	√	√	√	√	√	√		√			√
rhoCentralFoam	√	√	√	√							
rhoPimpleFoam	√	√	√	√					√		√
rhoSimpleFoam		√	√	√							√
scalarTransportFoam	√										
simpleFoam											√
spratFoam	√	√	√	√	√	√		√			√
XiFoam	√	√	√	√		√					√

Table 3: application solver capability matrix*

*: taken from: https://www.openfoam.com/documentation/guides/latest/doc/openfoam-guide-applications-solvers.html#sec-applications-solve

3 Conclusion

"I always thought something was fundamentally wrong with the universe" [1]

Figure 1: The Universe

A Gauss Theorem

$$\int_V (\nabla \cdot u) dV = \tag{11}$$

References

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy.* San Val, 1995.