

# OpenFOAM: An Introduction

**Hrvoje Jasak**

`h.jasak@wikki.co.uk`

**Wikki Ltd, United Kingdom**

**FSB, University of Zagreb, Croatia**

**18/Nov/2005**

Objective: A high-level overview of OpenFOAM

- Present a novel way of handling software implementation in numerical mechanics

Topics

- Requirements on modern CFD software
- A new approach to model representation
- Mesh handling and polyhedral support
- Pre-implemented capabilities

## State of the Art

- Numerical modelling part of product design
  - Improvements in computer performance
  - Improved physical modelling and numerics
  - Sufficient validation and experience
- Two-fold requirements
  - Models for wide area of physics + coupling
  - Complex geometry, high-performance computing, automatic meshing *etc.*

## Requirements on Software Design

- Industrial Environment
  - Integration into CAD-based process
  - Complex geometry and automatic meshing
  - Robust, fast and accurate solvers
- Research Organisations
  - Quick and reliable model implementation
  - Experimentation with various model forms
  - Separation between physics and numerics

How to handle complex models in software?

- Natural language of continuum mechanics: partial differential equations

$$\frac{\partial k}{\partial t} + \nabla \cdot (\mathbf{u}k) - \nabla \cdot [(\nu + \nu_t) \nabla k] = \nu_t \left[ \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T) \right]^2 - \frac{\epsilon_o}{k_o} k$$

- Main object = **operator**, e.g. time derivative, convection, diffusion, gradient

# OpenFOAM: CCM in C++



OpenFOAM (Field Operation and Manipulation):  
Represent equations 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)
);
```

# Object Orientation

Recognise main objects from the numerical modelling viewpoint

- Computational domain

Object	Software representation	C++ Class
Time	Time steps (database)	time
Tensor	(List of) numbers + algebra	vector, tensor
Mesh primitives	Point, face, cell	Point, face, cell
Space	Computational mesh	polyMesh

# Object Orientation

- Field algebra

Object	Software representation	C++ Class
Field	List of values	Field
Boundary condition	Values + condition	patchField
Dimensions	Dimension Set	dimensionSet
Geometric field	Field + boundary conditions	geometricField
Field algebra	+ - * / <i>tr()</i> , <i>sin()</i> , <i>exp()</i> ...	field operators

- Matrix and solvers

Object	Software representation	C++ Class
Linear equation matrix	Matrix coefficients	IduMatrix
Solvers	Iterative solvers	IduMatrix::solver

# Object Orientation

- Numerics

Object	Software representation	C++ Class
Interpolation	Differencing schemes	interpolation
Differentiation	ddt, div, grad, curl	fvc, fec
Discretisation	ddt, d2dt2, div, laplacian	fvm, fem, fam

Implemented Methods: Finite Volume, Finite Element, Finite Area and Lagrangian tracking

- Top-level organisation

Object	Software representation	C++ Class
Model library	Library	turbulenceModel
Application	main()	–

# Model Interaction

## Common Interface for Related Models

```
class turbulenceModel
{
    virtual volTensorField R() const = 0;
    virtual fvVectorMatrix divR
    (
        volVectorField& U
    ) const = 0;
    virtual void correct() = 0;
};
class SpalartAllmaras : public turbulenceModel{};
```

## Model-to-Model Interaction

```
fvVectorMatrix UEqn
(
    fvm::ddt(rho, U)
    + fvm::div(phi, U)
    + turbulence->divR(U)
    ==
    - fvc::grad(p)
);
```

New components do not disturb existing code

# Run-Time Selection

- Model-to-model interaction through common interfaces (virtual base classes)
- New components do not disturb existing code
- Run-time selection tables: dynamic binding
- Used for every implementation: “user-coding”
  - Convection differencing schemes
  - Gradient calculation
  - Boundary conditions
  - Linear equation solvers
  - Physical modelling, e.g. evaporation model, *etc.*

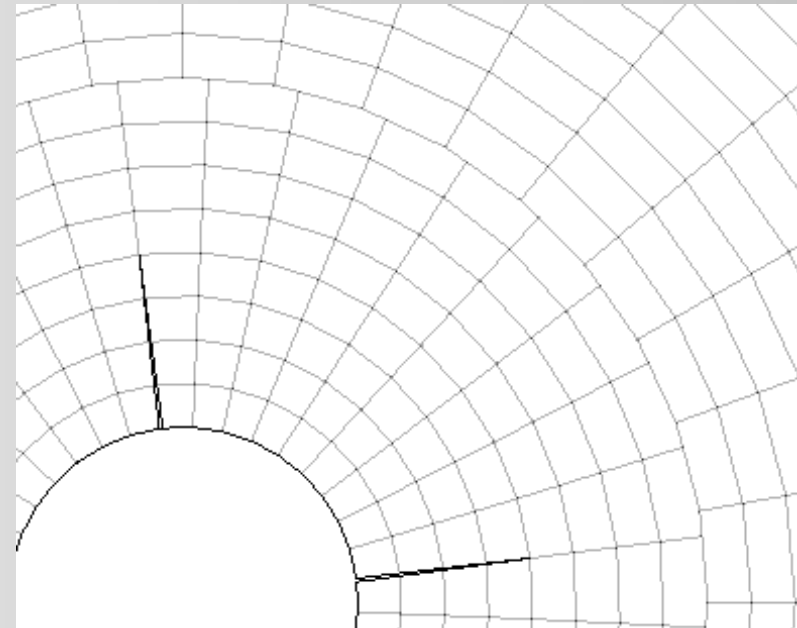
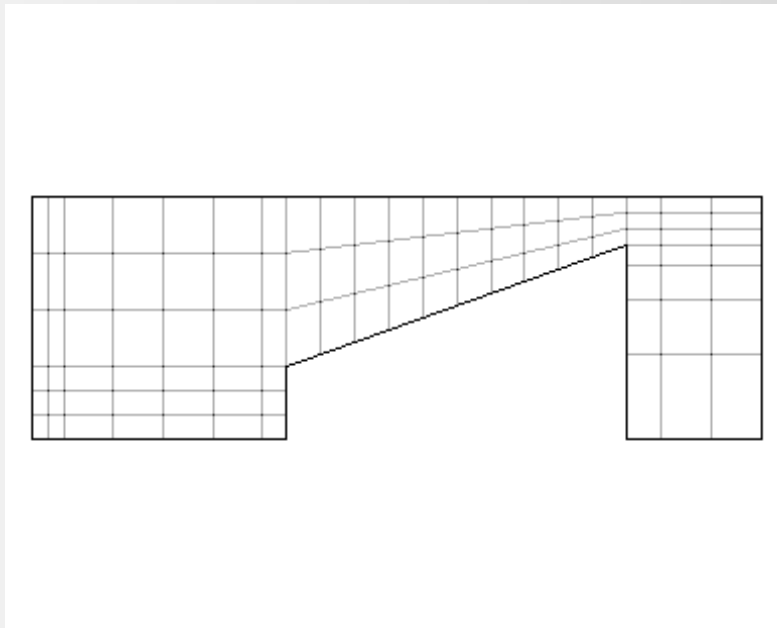
## Complex Geometry: Meshing Issues

- Complex geometry is a rule, not exception
- Polyhedral cell support
  - Cell is a polyhedron bounded by polygons
  - Consistent handling of all cell types
  - More freedom in mesh generation
- Automatic motion solver + topo morph engine

# Geometry Handling

## Time-Varying Geometry

- Automatic mesh motion solver
- Topological mesh changes with poly support



# Speed of Execution

## Handling large-scale computations

- Efficient numerics
  - Best discretisation practice for a given problem
  - Iterative solvers almost inevitable
  - Careful analysis of non-linearity and inter-equation coupling
- Massive parallelism: domain decomposition

# Layered Development



- Design encourages code re-use: shared tools
- Code developed and tested in isolation
  - Vectors, tensors and field algebra
  - Mesh handling, refinement, topo changes
  - Discretisation, boundary conditions
  - Matrices and solver technology
  - Physics by segment
  - Custom applications
- **Ultimate user-coding capabilities!**

# Implemented Capabilities



## Discretisation

- Polyhedral Finite Volume Method with moving mesh support (second and fourth-order)
- Finite Element Method on polyhedral cells
- Finite Area Method (2-D FVM on a surface)
- Lagrangian particle tracking model
- Ordinary differential equation solver

# Implemented Capabilities



## Model and Utility Libraries

- Thermo-physical models (liquids and gasses)
- Chemical properties
- Non-Newtonian viscosity models
- Turbulence models (RANS and LES)
- Dynamic mesh and topology changes
- A-posteriori error estimation
- Diesel spray (atomisation, dispersion, heat transfer, evaporation, spray-wall *etc.*)

# Implemented Capabilities



## Top-Level Solvers

- Top-level applications: provided with OpenFOAM or developed for the purpose
- Should be considered as both
  - Pre-packaged functionality
  - Examples of library use
- ... in short, this is a small subset of all solvers implemented in OpenFOAM

# Implemented Capabilities



## Top-Level Solvers

- Basic: Laplace, potential flow, transport
- Incompressible flow, compressible flow
- Heat transfer: buoyancy-driven flows
- Multiphase: Euler-Euler, surface capturing and tracking
- DNS and LES turbulent flows
- Combustion, spray and in-cylinder flows
- Stress analysis, electromagnetics, MHD, *etc.*

# Summary

- Object-oriented approach facilitates model implementation: layered design + re-use
- Equation mimicking opens new CCM grounds
- Extensive capabilities already implemented
- Open design for easy user customisation

## Acknowledgements and Further Info

- For more info on OpenFOAM, please visit <http://www.openfoam.org>
- OpenFOAM resources: <http://www.foamcfd.org>
- OpenFOAM User Group: <http://openfoam.cfd-online.com>
- OpenFOAM Workshops and Seminars

# FOAM: CCM in C++

## Main characteristics

- Wide area of applications: all of CCM!
- Shared tools and code re-use

## Versatility

- Unstructured meshes, automatic mesh motion + topological changes
- Finite Volume, Finite Element, Lagrangian tracking and Finite Area methods
- Efficiency through massive parallelism