OpenFOAM Tutorials: Basic Session

Hrvoje Jasak

h.jasak@wikki.co.uk

Wikki Ltd, United Kingdom and
FSB, University of Zagreb, Croatia

7-9th June 2007
Open Source CFD Platform

OpenFOAM: Open Source Computational Continuum Mechanics

- Commercial CFD vendors do not provide flexibility for customisation and add-on developments to answer the needs of commercial and research users
- Proprietary approach to solution methodology is the limiting factor: closed software architecture; no customer-specific proprietary models etc.
- Complete CFD methodology is already in the public domain (research papers, model formulation, numerical schemes, linear equation solvers etc.)
- **The way forward** involves educated users and customised simulations
- Objective: open source implementation of existing knowledge and an object-oriented platform for easy future development
  1. Completely open software platform using object-oriented design
  2. Extensive modelling capabilities in library form: component re-use
  3. Fast, robust and accurate numerical solver
  4. State of the art complex geometry handling
  5. On-order targeted and customer-driven model development
- ...but the mode of operation changes considerably
Implementing Continuum Models

How to Handle Complex Continuum Models in Software?

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

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

- Objective: represent differential equations in their natural language

```cpp
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 the implementation and the original equation is clear
Object Orientation

Object-Oriented Software

- Analysis of numerical simulation software through object orientation: “Recognise main objects from the numerical modelling viewpoint”
- Objects consist of data they encapsulate and functions which operate on the data

Example: Sparse Matrix Class

- Data members
  - Sparse addressing pattern (CR format, arrow format)
  - Diagonal coefficients, off-diagonal coefficients
- Operations on matrices or data members
  - Matrix algebra operations: +, −, *, /,
  - Matrix-vector product, transpose, triple product, under-relaxation
- Actual data layout and functionality is important only internally: efficiency

Example: Linear Equation Solver

- Operate on a system of linear equations $Ax = b$ to obtain $x$
- It is irrelevant how the matrix was assembled or what shall be done with solution
- Ultimately, even the solver algorithm is not of interest: all we want is new $x$!
- Gauss-Seidel, AMG, direct solver: all answer to the same interface
Object Orientation

Basic Components

- Scalars, vectors and tensors with algebra
- Computational mesh; mesh motion, adaptive refinement, topological changes
- Fields (scalar, vector, tensor) and boundary conditions: Dirichlet, Neumann etc.
- Sparse matrix support with linear solver technology

Discretisation Classes

- Implemented as interpolation, differentiation and discretisation operators
- All discretisation methods use identical basic components, e.g. common mesh and matrix support. Better testing and more compact software implementation

Physical Modelling Libraries and Top-Level Solvers

- Libraries encapsulate interchangeable models with common interfaces
- New models provide functionality adhering to common interface
- Custom-written and optimised top-level solvers for class of physics

Utilities

- Common functionality is needed for most simulation methods
- Example: problem setup, mesh manipulation, data analysis
Running OpenFOAM

1. Laminar fluid flow in a lid-driven cavity
   - Pre-processing with FoamX
   - Case structure and dictionary format
   - Running OpenFOAM
   - Basic post-processing
   - Create a graded mesh, map a solution and repeat a simulation

2. Free surface flow: dam break problem
   - Pre-processing using file editors and command line
   - Multi-block mesh generation with `blockMesh`
   - Setting non-uniform initial field
   - Post-processing: create an animation
   - Running OpenFOAM on multiple processors
Flow in a Lid-Driven Cavity: Case Setup

\[ U_x = 1 \text{ m/s} \]

\[ d = 0.1 \text{ m} \]
Mesh Generation: blockMesh
Lid-Driven Cavity

Dimensions and Properties
- Domain size: $0.1 \times 0.1$ m
- Lid velocity: $1$ m/s
- Kinematic viscosity: $0.01$ m$^2$/s

Simulation Settings
- Start time = $0$
- End time = $0.5$ s
- Time step: $\Delta t = 0.005$ s

Sampling Lines
- $(0.0500.0005)$ to $(0.050.10.0005)$
- $(0.0500.0005)$ to $(0.050.10.0005)$
Lid-Driven Cavity

Phases of Simulation
- Mesh generation
- Boundary conditions
- Discretisation control
- Running the simulation
- Graphical post-processing
- Velocity magnitude
- Sampling lines

Mesh Refinement
- Mesh generation
- Solution mapping
- Running the simulation
- Graphical post-processing
Structure of an OpenFOAM Case

- Unlike other applications, mesh, case setup and results data in OpenFOAM is split into a number of files.
- The case is stored in a directory, with a prescribed structure.
- A case consists of:
  - Mesh or series of meshes (deformation, topological changes)
  - Material properties
  - Choice of models
  - Solver settings: discretisation choice, convergence tolerances, etc.
  - Results of simulation, organised in time or iteration directories
- Idea: keep similar cases together: root directory
- Case data separated into units: case directory
- Typically, 1 high-level object lives in 1 file
- Mesh separated into several files: point motion/topological change
Case Data Format and Organisation

Data File Format

- All OpenFOAM files annotated by a version-control and context header
- Header contains
  - File version, format and location (root/cas/instance)
  - Class type and object name of the class the data belongs to
- C++ comment style applies to all parsed data

```cpp
FoamFile
{
    version 2.0;
    format ascii;

    root "";
    case "";
    instance "";
    local "";

    class dictionary;
    object transportProperties;
}
```
Dictionary Format

- Main input system: dictionary
  - List of keyword – value pairs delimited by semicolon
  - Order of entries is irrelevant
  - Embedded dictionary levels allowed
  - Field data also in dictionary format: uniform or expanded data, using “supertokens” for efficient reading of expanded data

- Embedded dictionaries use curly brackets delimitation
- Main dictionary, controlling execution: `system/controlDict`
- All other dictionaries tailored to applications
- Example: transport properties for `icoFoam`

```plaintext
nu nu [0 2 -1 0 0 0 0] 0.01;
```
Dictionary Format: Field Data Example

dimensions [0 1 -1 0 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
    movingWall
    {
        type fixedValue;
        value uniform (1 0 0);
    }
    fixedWalls
    {
        type fixedValue;
        value uniform (0 0 0);
    }
    frontAndBack
    {
        type empty;
    }
}
OpenFOAM Case Data Format and Organisation

OpenFOAM Case Structure

- `<case>`
  - `system`
    - `controlDict`
    - `fvSchemes`
      - `fvSolution`
    - `constant`
      - `… Properties`
      - `polyMesh`
        - `points`
        - `cells`
        - `faces`
        - `boundary`
  - time directories
Dam Break Free Surface Flow

Water column: 0.292 m

Height: 0.584 m

Volume: 0.048 m

Width: 0.146 m

Depth: 0.024 m
Dam Break Free Surface Flow

Material Properties

- $\rho_1 = 1000 \text{ kg/m}^3$
- $\rho_2 = 1 \text{ kg/m}^3$
- $\nu_1 = 1 \times 10^{-6} \text{ Pa/s}$
- $\nu_2 = 1.48 \times 10^{-6} \text{ Pa/s}$
- Gravity vector $(0, -9.81, 0) \text{ m/s}^2$

Initial field

- Column of water, 0.292 m high and 0.1461 m wide
- Both fluids at rest at $t = 0 \text{ s}$
OpenFOAM: Implemented Capabilities

Discretisation Methods and Support
- Second and fourth-order Finite Volume with mesh motion and topological changes
- Polyhedral Finite Element solver (mesh motion)
- Lagrangian particle tracking (discrete particle model); Diesel spray model
- Finite Area Method: FVM on a curved surface in 3-D
- A-posteriori error estimation
- Dynamic mesh handling and topology changes; automatic mesh motion

Physical Modelling Libraries
- Thermo-physical models (liquids and gasses)
- Chemical reaction library interface (Chemkin)
- Non-Newtonian viscosity models
- Turbulence models (RANS and LES, compressible and incompressible); DNS
- Diesel spray (atomisation, dispersion, heat transfer, evaporation, spray-wall etc.)

High Performance Computing Support
- Massively parallel computing: domain decomposition approach
- Next-generation of linear equation solver technology: up to 3 times faster!
OpenFOAM: Implemented Capabilities

Standard Top-Level Solvers

- Basic: Laplace, potential flow, transport
- Incompressible flow, compressible flow
- Heat transfer: buoyancy-driven flows
- Multiphase: Euler-Euler, surface capturing and surface tracking
- DNS and LES turbulent flows, aero-acoustics
- Pre-mixed and Diesel combustion, spray and in-cylinder flows
- Stress analysis, fluid-structure interaction, electromagnetics, MHD, etc.

Utilities

- Pre-processing, data manipulation
- Mesh import and export, mesh generation and manipulation
- Parallel processing tools: decomposition and reconstruction
- Post processor hook-up (reader module) and data export

This is just a “standard set”: users write their own applications using the library