A Year in the Life of OpenFOAM

Hrvoje Jasak

h.jasak@wikki.co.uk, hrvoje.jasak@fsb.hr

Wikki Ltd, United Kingdom, Germany and Australia
Faculty of Mechanical Engineering and Naval Architecture
University of Zagreb, Croatia

6th OpenFOAM Workshop, Penn State University 13-16 June 2011
Topics

- Introduction
- OpenFOAM on Microsoft Windows
- OpenFOAM Turbo Tools: mixing plane interface
- Improvements in compressible flows
  - New density-based solver: dbnsFoam
  - Multi-phase compressibility: under-water explosions
- Parallel dynamic mesh simulations: capability upgrade
- Shape optimisation and adjoint in OpenFOAM
- OpenFOAM on GPU
- Project management and code deployment: problems and challenges
- Summary and Outlook
OpenFOAM and the Summer of 2011

- It feels like the hot summer of 1969: Love is All Around Me!
- Today, OpenFOAM is a proven player in commercial CFD and academic research
- Project work-load is impressive and shows a new way forward: customers organise in consortia to share development cost and experiences in migration
- Biggest problems quoted by the community
  - **Steep learning curve for new users**
  - **Insufficient documentation**: capability is there but we do not know how to set up cases and use it to the full potential
  - Need to make validation examples publicly accessible
  - Concerted effort on open source graphical user interface
  - (Personal problem): Everybody seems to want to use the capability I am currently working on, even before it is ready
- We need to **grow the expert-level knowledge in the community**
  - 16-session OpenFOAM training at OFW-6: Congratulations!
  - NUMAP-FOAM 2011, 4th edition of Summer School
  - Other hosted events for expert users?
Native Microsoft Windows Version of OpenFOAM

- Objective: complete bottom-up Windows-compliant build and deployment
- Using native compiler and operating system interface (non-POSIX), within Microsoft Visual Studio development environment
- New mechanism for include file handling: Michael Wild, FreeFOAM
- Library symbol import-export handling: changes to all class files!

Remaining Items

- Rewrite of static debug switches for easier deployment
- HPC tuning with Windows HPC 2008
- Merge with main development line (test of integration with Debian packages)

Results: **Test Deployment with a Commercial Client Under Way (June 2011)**
Turbomachinery Simulations in OpenFOAM

- Work driven by Turbomachinery SIG very impressive: incompressible turbomachinery CFD capability completely covered
- Special thanks to Martin Beaudoin, Hakan Nilsson, Maryse Page
- **General Grid Interface (GGI)** and its derived forms
  - Cyclic GGI interface
  - Partial overlap GGI, contribution by Oliver Borm
  - Mixing plane interface
- Upgrade on speed of parallel execution due for release: improved polygon search (and intersection?) algorithm, upgrade in the parallel GGI communications pattern
Mixing Plane Interface

- Turbomachinery components include multiple identical channels: modelling a single channel simplifies the simulation
- **Banded circumferential averaging** at rotor-stator interface allows the transient problem, with substantial reduction in simulation cost
- Circumferential averaging introduces a one-time mixing loss, but without other adverse effects: blade wakes and passage-to-passage variation averaged out
- Consistency in interpolation is essential for mass conservation and accuracy!
Implementation of a Mixing Plane Interface

- Basic GGI interpolation tool already performs the correct operation
  - Area-weighted interpolation: can be done in ribbons
  - Strict mass conservation: banded patches to achieve circumferential averaging in a prescribed manner

- Implementation: mixing plane = **two GGI interfaces used back-to-back**
- (The more we work on this, the easier it becomes: no interpolation bias)
- Mixing plane under testing: Martin Beaudoin, Hydro Quebec, AJK 2011 paper
Compressible Solver: dbnsFoam

New Compressible Flow Solver: dbnsFoam

- Currently, OpenFOAM is not competitive for compressible flow with strong compressibility effects: high speed, shocks. **Block-solution is required!**
- dbnsFoam solver utilises **flux vector splitting** (HLLC and derivatives), flux difference splitting (Roe) and central (Local Lax-Friedrichs) schemes; multidimensional limiters are available
- Time stepping schemes: multistage and pseudo-time stepping algorithms
- Fully integrated GGI and MRF capabilities: interfacing with Turbo Tools
- FAS multigrid under testing; Implicit solution planned in the near term
- Currently focused on turbomachinery and external aerodynamics - planning for reactive flows and frequency-based methods
Multi-Phase Compressibility

Simulation of Under-Water Explosions

- This is ongoing collaboration with Johns Hopkins APL, Penn State University and Wikki: working hard for more than a year
- Dominating effects of compressibility in air and water: massive change in density, with propagating pressure waves
- Pressure ranges from 500 bar to 20 Pa
- Stiff numerics: collapse of over-expanded bubble due to combined compressibility of both phases are the basis of the phenomenon
- Test cases: Rayleigh-Plesset oscillation, undex under a plate, explosion
- Eric Paterson, Scott Miller, David Boger, Penn State; Ashish Nedungadi, JHU-APL
Multi-Phase Compressibility

First Simulations of Under-Water Explosions: Eric Paterson, Penn State

- Bubble of high initial pressure expands after explosion
- Initial pressure pulse is very fast - with little effect
- Bubble collapse creates re-entrant jet which pierces the free surface
- Stability problems resolved in segregated solver
- Next phase: block-coupled $p - \alpha$ solution algorithm
Parallelisation of Topological Changes

Topologically Changing Meshes in Parallel Simulations

- Topology engine did not account for parallelisation: synchronisation and triggering of topological changes
- For good scaling with parallel execution
  - Mesh change must be localised on a single processor
  - ...but topology update must be synchronised!
- Different types of topology modifiers behave in a different manner
- **Domain reconstruction completely broken**: no addressing data
- Issue of dynamic load balancing on topologically changing meshes becomes acute
Layer Addition-Removal

Parallelisation of Layer Addition-Removal Mesh Modifier

- `layerAdditionRemoval` mesh modifier removes cell layers when the mesh is compressed and adds cells when the mesh is expanding. Definition:
  - Oriented face zone, defining an internal surface
  - Minimum and maximum layer thickness in front of the surface
  - Both internal and patch faces are allowed

```plaintext
right
{
    type layerAdditionRemoval;
    faceZoneName rightExtFaces;
    minLayerThickness 0.0002;
    maxLayerThickness 0.0005;
    active on;
}
```

Tasks

- Allow face zone to be distributed over multiple processors
- Synchronise logic for addition and removal: interface is formally present on all processors, even if it locally has zero faces
- Check and validate decomposition normal to face zone: mesh motion issues
Parallelisation of Sliding Interface Mesh Modifier

- `slidingInterface` allows for relative sliding of components. Definition:
  - A master and slave patch, originally external to the mesh
  - Allows uncovered master and slave faces to remain as boundaries

```foam
mixerSlider
{
    type slidingInterface;
    masterPatchName outsideSlider;
    slavePatchName insideSlider;
    projection visible;
    active on;
}
```

Tasks

- Sliding surfaces are defined as a patch pair and allow partial coverage
- No global logic: sliding pair may be local on a processor mesh
- Protection in parallel decomposition is required for points: this is a task for domain decomposition tool
Parallelised Topological Changes

Low-Level Work in Parallelised Topological Changes

- New effects to be accounted for
  - Topological change happening on some processors but not on others
  - Re-calculation of local and global mesh data involves communication which need to be synchronised when topological change is local
  - Synchronisation required for processor boundary faces that undergo a topological change: identical operation on both processors
- Data mapping on neighbouring processor may require local update for parallel synchronisation
- Currently addressing issues of load balancing on a problem-by-problem basis
Parallel Decomposition for Engines

Example: engineScotch Domain Decomposition Method in Motion
Domain Reconstruction Tool for Topologically Changing meshes

- Standard decomposition tools use point/face/cell/boundary maps between a single CPU and processor mesh. Maps are created on a static mesh with decomposition.
- With parallelised topological changes, this breaks down completely: global mesh and numbering do not exist and cannot be implied.
- Solution: use processor meshes to build a global mesh from scratch, by adding processor meshes in order, merging shared points and faces.

reconstructParMesh Utility

- In presence of maps (no topological changes) use standard method.
- Upon topological changes build and merge the mesh, adding cells in order of processor index and assemble mapping data.
- Fields on reconstructed mesh can be assembled or decomposed as before.
- -cellDist option can be used to visualise domain decomposition: this time it is assembled on reconstruction!
Shape Optimisation and Adjoint

Types of Optimisation

- **Deterministic Methods**: all observables and controls assumed deterministic
- **Stochastic Optimisation**: Probability density function associated with all inputs and controls

Types of Analyses: Gradient or Non-Gradient Based Analysis

- Sensitivity analysis
- Uncertainty analysis: robust design
- Parameter estimation
- Geometric shape optimisation; Topology optimisation

Software Toolkit

1. **Low-degree shape parametrisation**: RBF morphing, (free form deformation)
2. **Direct mesh representation and deformation**: automatic mesh motion solver
3. **CFD solver and evaluation of objective**: OpenFOAM solvers
4. **Optimisation algorithm**: native or external optimiser
   - Non-gradient based: native Simplex Nelder-Mead
   - Gradient-based: Newton-type, using tangent and adjoint (limited capability with scope for further work)
Shape Optimisation and Adjoint

Gradient-Based Optimisation in OpenFOAM

- Options on formulation of derivative: **Tangent method, Adjoint solver**
- Options on evaluation: **discrete or continuous method**
- Currently implemented only a laminar continuous adjoint solver. This is not publicly available and requires rewrite of numerics, using block coupling
- Prototype implementation for **automatic differentiation** (discrete method): FadOne class and operator overloading

Continuous Adjoint Optimisation for Turbulent Flows in OpenFOAM

- Solving adjoint incompressible Navier-Stokes system with a complete turbulence model with near-wall treatment
  - Adjoint momentum and continuity equation
  - Adjoint $k − \epsilon$ turbulence model with adjoint wall treatment
- This is a specific tool for specific problems: derivation for internal and external flows is different (eg. surface forces vs. pressure loss)
- Substantial mathematical effort required: boundary conditions are derived specifically for the chosen objective
Automatic Differentiation

Background on Forward Derivatives

- In CFD we specify a number of input variables (mesh, inlet velocity, attack angle), perform a series of mathematical operations to get some output (pressure distribution, lift coefficient).

- Clearly, all output variables depend on the input and mathematics is tractable.

- Therefore, forward derivatives, propagation of uncertainty etc. simply describes how the output data depends on the input: we wish to recover this dependence not only in value but also in derived properties.

Implementation

- In all cases, implementation provides a discrete forward derivative and discrete adjoint: full consistency with the physical model and discretisation.

- Operation on derivatives (or other attached data) performed as side-effects on basic operations: templating on scalar type.

- Currently propagated to matrix level, excluding the mesh module.

- For shape optimisation, parametric geometry and mesh is required.

- Aim: pull through the sensitivity analysis machinery through complete OpenFOAM using templating trickery and meta-programming.
OpenFOAM on GPU

- Presenting work by Qingfeng (Jason) Xia, Manchester University (UK)
- Other implementations also exist, based on CUDA 3rd party libraries
- Implementation
  - Work is based on OpenCL, which can be executed both on GPU and CPU; NVIDIA and ATI graphics cards are supported
  - Implementation of CG and BiCG solvers from base-up, using lduMatrix class and rewrite of Krylov Subspace solvers
  - Looking for an OpenCL capable GPU; support for double precision
- clFoam single precision tested on ATI 5650M GPU and NVIDIA Tesla C2050
- Problems and challenges
  - Solution of the linear system on the GPU requires massive migration of data: substantial impact on performance
  - Parallelisation and GPU: two-levels of data exchange
  - Is it feasible to migrate the complete matrix assembly to GPU? What about data structures and efficiency of indirect addressing access
  - Work-in-progress – not ready for release
- GPU programming hurts: need a better interface and more control!
Deployment and Release

Resolving Issues of Copyright, Trademark and License

- OpenFOAM is now an established community-driven project: need to create reliable community-driven infrastructure
- Resolution of Trademark and Copyright issues under way: priority is to serve the needs of the Open Source development framework
- Working towards formal setup of the OpenFOAM Foundation

Improvement in the Release Cycle

- Some successes
  - Documentation and community portal: working well!
  - Automated public test loop, nightly builds, validation cases
  - Integrated cross-platform version - Windows merge under way
- ... and some problems
  - Failed on release schedule: basically a HJ work-load problem
  - Integration of new features is too slow!
  - Need better validation cases and best practice guidelines aimed at attracting new users and breaking down barriers on code reliability
A Year in Life of OpenFOAM

- We are the flavour of the moment!
- This is the time to formalise the community organisation

Outlook and Wish List

- Better and more regular release schedule: release committee and more hands on deck!
- Formalise and validate software capability: adding new features is as important as making the best possible use of existing capability
- Expand the pool of expert developers: NUMAP-FOAM Summer School, Zagreb, but also other events, eg. OpenFOAM Summer of Code?

- **OpenFOAM Workshop in 2012**: Europe, USA, Far East or Down Under?

**Eye Candy: Evaporation Kernel Growth**, Christian Kunkelmann, TU Darmstadt