Multi-Physics Simulations in Continuum Mechanics

Hrvoje Jasak
h.jasak@wikki.co.uk

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

Objective

- Examine implementation framework for coupled fluid-structure simulations
  - Implementing continuum models in software
  - Complex geometry support: moving-deforming mesh
  - Coupling algorithms for fluid-structure problems

Topics

- Background: Numerical simulation in engineering
- Software architecture and limitations
- Object orientation is used: mimicking partial differential equations in code
- Complex geometry support: moving deforming mesh and topological changes
- Handling of fluid-structure coupling
- Example simulations
- OpenFOAM: Open Source numerical simulation software
Established Tool for Engineering Design

- Numerical simulations of discrete and continuous systems is established in engineering use
- In some cases, simulation has completely displaced experiments
  - Aerospace: Airbus 380
  - Automotive components: under-hood, passenger compartment, acoustics
  - Electronics cooling
  - Hazard simulations, explosions etc.
- Moving beyond traditional structural analysis and fluid flow
  - Electromagnetics
  - Magneto-hydrodynamics
  - Complex coupled heat-mass transfer phenomena
- Massive increase of available computing power opens new horizons
  - Industrial Large-Eddy Simulation
  - Physics-based models replacing correlations
  - Multi-dimensional modelling in place of 1-D equations
  - Coupled simulation of interacting physics: fluid-structure interaction
State of the Art in Commercial CFD

- Trend towards unification of numerical tools, especially structural analysis + CFD
- Two-fold requirements
  - Integration into the CAD-based design process
  - Quick and reliable implementation of new models
- Complex geometry support, high-performance computing, automatic meshing, dynamic mesh capabilities etc. needed across the spectrum

Software Organisation

- A CFD software is a large project: order of 1-2 million lines of source code
- Large number of physical models: complex model-to-model interaction
- Functional approach: centralised data and functions operating on it
- Single discretisation method (FVM, FEM) associated with physics
- Monolithic implementation and integrated software: single executable for all cases

Consequences

- User-defined models inefficient and limiting; at worst, impossible to implement
- **Difficulties in development, maintenance and support**
Implementing Continuum Models

How to Handle Complex Continuum Models in Software?

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

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

- 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 coefficient
  - Off-diagonal coefficients

- Operations on matrices or data members
  - Matrix algebra: +, −, *, /, scalar multiplication
  - Matrix-vector product, transpose, triple product, under-relaxation

Example: Linear 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

### Primitive Objects: Space and Time

<table>
<thead>
<tr>
<th>Object</th>
<th>Software representation</th>
<th>C++ Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensor</td>
<td>(List of) numbers + algebra</td>
<td>vector, tensor</td>
</tr>
<tr>
<td>Mesh primitives</td>
<td>Point, face, cell</td>
<td>point, face, cell</td>
</tr>
<tr>
<td>Space</td>
<td>Computational mesh</td>
<td>polyMesh</td>
</tr>
<tr>
<td>Time</td>
<td>Time steps (database)</td>
<td>time</td>
</tr>
</tbody>
</table>

### Field Algebra

<table>
<thead>
<tr>
<th>Object</th>
<th>Software representation</th>
<th>C++ Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>List of values</td>
<td>Field</td>
</tr>
<tr>
<td>Boundary condition</td>
<td>Values + condition</td>
<td>PatchField</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Dimension set</td>
<td>dimensionSet</td>
</tr>
<tr>
<td>Geometric field</td>
<td>Field + mesh + boundary conditions</td>
<td>GeometricField</td>
</tr>
<tr>
<td>Field algebra</td>
<td>$+ - */\text{tr}, \sin, \exp \ldots$</td>
<td>field operators</td>
</tr>
</tbody>
</table>
Matrix Algebra and Linear Equation Solvers

<table>
<thead>
<tr>
<th></th>
<th>Software representation</th>
<th>C++ Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear equation matrix Solvers</td>
<td>Matrix coefficients</td>
<td>IduMatrix</td>
</tr>
<tr>
<td></td>
<td>Iterative solvers</td>
<td>IduMatrix::solver</td>
</tr>
</tbody>
</table>

Numerics: Discretisation Methods

<table>
<thead>
<tr>
<th></th>
<th>Software representation</th>
<th>C++ Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation</td>
<td>Differencing schemes</td>
<td>interpolation</td>
</tr>
<tr>
<td>Differentiation</td>
<td>ddt, div, grad, curl</td>
<td>fvc, fec</td>
</tr>
<tr>
<td>Discretisation</td>
<td>ddt, d2dt2, div, laplacian</td>
<td>fvm, fem, fam</td>
</tr>
</tbody>
</table>

Revisiting the Equation

```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)
);```
Model-to-Model Interaction

Common Interface for Model Classes

- Physical models grouped by functionality, e.g. material properties, viscosity models, turbulence models etc.
- Models answer to common interface: new model does not disturb existing code

```cpp
class turbulenceModel
{
    virtual volTensorField R() const = 0;
    virtual fvVectorMatrix divR
    (
        volVectorField& U
    ) const = 0;
};

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

OpenFOAM Software Architecture

- Design encourages code re-use: developing shared tools
- Classes and functional components developed and tested in isolation
  - Vectors, tensors and field algebra
  - Mesh handling, refinement, mesh motion, topological changes
  - Discretisation, boundary conditions
  - Matrices and linear solver technology
  - Physics by segment in library form
  - Custom-written top-level solvers optimised for efficiency and storage
  - Library level mesh, pre-, post- and data handling utilities
- Custom-written top-level solvers optimised for efficiency and storage
- Development of model libraries: easy model extension
- Model-to-model interaction handled through common interfaces
- New components do not disturb existing code: fewer bugs
- Run-time selection provides ultimate user-coding capabilities: differencing schemes: convection, diffusion, rate of change; gradient calculation; boundary conditions; linear equation solvers; physical models; mesh motion algorithms etc.
Implemented Capabilities

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

Model and Utility 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
Linear Equation Solver

![Graph showing residual vs. iteration for various methods and classes. The graph includes lines for AMG, ICCG, Class 1, Class 2, A, Class 2, B, Class 2, C, and Non-diverging. The x-axis represents iteration, and the y-axis represents residual. The graph illustrates the convergence rates of different methods and classes.]
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 and data export

Complex Solvers and Applications

- The above is just a “standard set”: library users write own applications or combine the above. Example: fluid-structure interaction solvers
Complex Geometry Handling

• Complex geometry is a rule, not exception
• Polyhedral cell support: first in class
• Interfaces to all major mesh generators

Automatic Mesh Motion Solver

• Simulation with variable domain shape
• Based on the prescribed boundary deformation, re-calculate the point position
• Handles solution-dependent boundary deformation, e.g. contact problem of elastic solids

Topological Changes

• For cases with considerable mesh deformation
• Cell addition/removal; sliding interfaces
• Multiple interacting topological operation for complex motion, e.g. opening valves
• Automatic handling of solution mapping
Fluid-Structure Interaction

Coupled Fluid-Structure Simulation

- A prerequisite for Fluid-Structure Interaction (FSI) is a “single-physics” solver
- Multi-physics capability in a single platform is beneficial
  - Multiple co-existing domains operating side-by-side
  - Choice of structural analysis and fluid flow models
  - Models share the same infrastructure: common mesh support and linear algebra classes provide freedom in coupling

Levels of Coupling

- **Explicit two-level coupling.** Simulations run side-by-side and exchange coupling information in an explicit manner. Formally performing Picard iterations, coupling fails even for modestly interacting problems

- **Matrix-level interaction.** Physical models are discretised separately and coupling is described in implicit manner. All components are combined into a single linear system before the solution, resulting in improved stability

- **Equation-level coupling.** Recognising common conservation equations, fluid-structure system is described as a single continuum. Choice of common primitive variable and reformulated governing laws provide closest possible coupling. A non-standard form of equation is ideally suited for OpenFOAM
Example: Flow-Induced Vibration

Flow-Induced Deformation

- Traditional explicit coupling: Picard iterations. Pressure transferred from fluid to structure and displacement from structure to fluid: profile and force conservation
- Fluid: incompressible flow model
- Stress analysis: linear response with large deformation
- Automatic mesh motion deforms the fluid mesh
- On solid side, mesh deformation is a part of the solution
Example: Falling Containers

Falling Plastic Containers

- At impact, complex flow field causes deformation of a solid: travelling wave in an elastic pipe. Wave speed dependent on properties of the coupled system
- Very strong interaction: mean flow and stress is zero. Explicit coupling fails
- Equation-level coupling: one system approach

\[ \frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho uu) = \nabla \cdot \left[ 2\eta \dot{\epsilon} - \frac{2}{3} \eta \text{tr}(\dot{\epsilon}) I - p I \right] \]

\[ \frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho uu) = \nabla \cdot \left[ 2N \dot{\epsilon} - \frac{2}{3} N \text{tr}(\dot{\epsilon}) I - p I \right] + \nabla \cdot \Sigma \]

where

\[ N = \mu \Delta t \quad p = -\frac{1}{3} \text{tr}(\sigma) \quad \Sigma = \int_{t=0}^{t-\Delta t} \left[ 2\mu \dot{\epsilon} - \frac{2}{3} \mu \text{tr}(\dot{\epsilon}) I \right] dt \]
Example: Falling Containers

- Combined experimental and numerical study
- Influence of bottle geometry: flat or curved bottom
- Further simulations include container fracture using a cohesive zone model for fracture dynamics
- As the crack opens, fluid within the container is exposed to atmospheric conditions and leaks out, changing the flow field

Results courtesy of dr. A Karač, University of Zenica and prof. A. Ivanković, UC Dublin
Free surface tracking

- 2 phases = 2 meshes
- Surface pressure, momentum and deformation is a part of the solution
- Coupling via kinematic and dynamic condition on free surface
- Mesh adjusted for deformation of the interface: solution-dependent motion
Free-Rising Air Bubble with Surfactants

Complex coupling problem

- FVM flow solver
- FEM automatic mesh motion
- FAM for surfactant transport
Open Source Simulation Tool

Open Source in Computational Continuum Mechanics

- Complete methodology is already in the public domain (research papers, model formulation, numerical schemes, linear equation solvers etc.)
- Objective: open source implementation of existing knowledge on an object-oriented platform for easy sharing and future development

Research with OpenFOAM

- Open architecture and extensive capabilities make a good research platform
- First OpenFOAM Workshop, Zagreb Jan/2006: 80 attendees from 3 continents
- Leading research/development centres: Chalmers University, Sweden; Politecnico di Milano, University College Dublin, TU Freiberg, Germany
- Major development on multi-phase flows: MFIX-NG NETL, US Dept. of Energy

OpenFOAM in Industry

- An open platform for in-house or specialist software development is required
- Interest greatly increased in the last year, following PhD projects, study visits or joint development projects. Migrating in-house knowledge on a common platform
- Pilot projects or active use on over 200 sites: numerous research centres (USA, Canada, Norway) and commercial companies
Summary

Fluid-Structure Interaction Simulations
- Maturing industrial environment needs unified tools and coupled simulations
- Functional software design of current generation tools limits applicability and causes difficulties in development and support
- Object-oriented technique: common platform for numerical simulation tools. Shared components, layered design and code-reuse
- New generation of solvers suited for coupled simulations: fluid-structure interaction

**OpenFOAM**: Open Source Platform for Numerical Simulations
- Equation mimicking opens new grounds in Computational Continuum Mechanics
- Extensive capabilities already implemented, including complex geometry support
- Open design for easy user customisation, including fluids and structures
- Solver customisation to problem class and physics
- High performance computing: massive parallelism, optimised solver technology
- Multiple discretisation methods sharing basic components for easy coupling

More info on OpenFOAM software and project
- [http://www.openfoam.org](http://www.openfoam.org)
- [http://www.foamcfd.org](http://www.foamcfd.org)