OpenFOAM Optimization Tools

Henrik Rusche and Aleks Jemcov

h.rusche@wikki-gmbh.de and a.jemcov@wikki.co.uk

Wikki, Germany and United Kingdom

Agenda

Objective

- Review optimisation problems
- Status in OpenFOAM

Topics

1. Introduction
2. Problem description and classification
3. What is OpenFOAM
4. Shape optimisation with radial basis functions
5. Gradient based optimisation techniques
6. Topology optimisation with adjoint equation
7. Summary

Introduction

- Find

$$min(f(p))$$

  where f is the objective functional
- f is calculated by executing a computational workflow (pre-processing, solver, post-processing)
- Classification:
  - Single vs. Multi-Objective problem (f is a vector)
  - Unconstraint vs. constraint problem (additional restrictions on f and/or p)
  - Parameter vs. Shape or Topology optimisation (geometry is changed)
  - Gradient vs. non-gradient based algorithms (uses $\frac{\partial f}{\partial p}$)
- Optimization as a discipline is very rich in methods, approaches and algorithms
- Many external tools exists (Dakota, modeFrontier, LMS Optimus)

Different Approaches

General approaches to optimization

- Non-gradient based algorithms
  - Simplex, Brent's, and Powell's methods
  - Genetic algorithms
  - Surrogate and reduced order model algorithms
- Gradient based algorithms
  - Steepest descent
  - Newton's method
  - Conjugate gradient
  - Sequential quadratic programming (SQP)
- Gradient calculation
  - Finite differences, incomplete gradient, reduced order model for gradients, etc.
  - Continuous adjoint and tangent equations
  - Discrete adjoint and tangent equations

OpenFOAM: Executive Overview

WIKKI

What is OpenFOAM?

- **OpenFOAM** is a free-to-use Open Source numerical simulation software with extensive CFD and multi-physics capabilities
- Free-to-use means using the software without paying for license and support, including **massively parallel computers**: free 1000-CPU CFD license!
- Software under active development, capabilites mirror those of commercial CFD
- Substantial installed user base in industry, academia and research labs
- Possibility of extension to non-traditional, complex or coupled physics: Fluid-Structure Interaction, complex heat/mass transfer, internal combustion engines, nuclear

Main Components

- Discretisation: Polyhedral Finite Volume Method, second order in space and time
- Lagrangian particle tracking, Finite Area Method (2-D FVM on curved surface)
- Massive parallelism in domain decomposition mode
- Automatic mesh motion (FEM), support for topological changes
- All components implemented in library form for easy re-use
- Physics model implementation through **equation mimicking**

Implementing Continuum Models

Equation Mimicking

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

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

- Objective: **represent differential 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)
);
```

- Correspondence between the implementation and the original equation is clear

Non-gradient Based optimization

- Non-gradient based optimization is based on the evaluation of gradients of objective functional in which OpenFOAM is responsible for providing the current state
- Usually several states are saved in order to deduce the behavior of the function being minimized
- This knowledge is used to retain and discard the states in an effort to find the optimal point
- Typical problem with non-gradient based algorithms is so called “curse of dimensionality” - too many parameters require excessive number of non-linear function evaluations thus making the algorithms very expensive
- In shape optimization effective geometrical parameterization is crucial in controlling the cost of the optimization algorithm

RBF Mesh Morphing and Motion

Radial Basis Function (RBF) Interpolation

- RBF interpolation defines the interpolation directly from the sufficient smoothness criterion on the interpolation (positive weighting factors):

$$s(\mathbf{x}) = \sum_{j=1}^{N_b} \gamma_j \phi(|\mathbf{x} - \mathbf{x}_{b,j}|) + q(\mathbf{x})$$

- Weighting factors calculated by inversion of a (small) dense matrix

RBF Interpolation Procedure

1. Establish locations of data-carrying points $\mathbf{x}_b$ and their values
2. Assemble and solve the equation set for γ and β using a direct solver
3. Calculate values at desired locations by evaluating $s(\mathbf{x})$

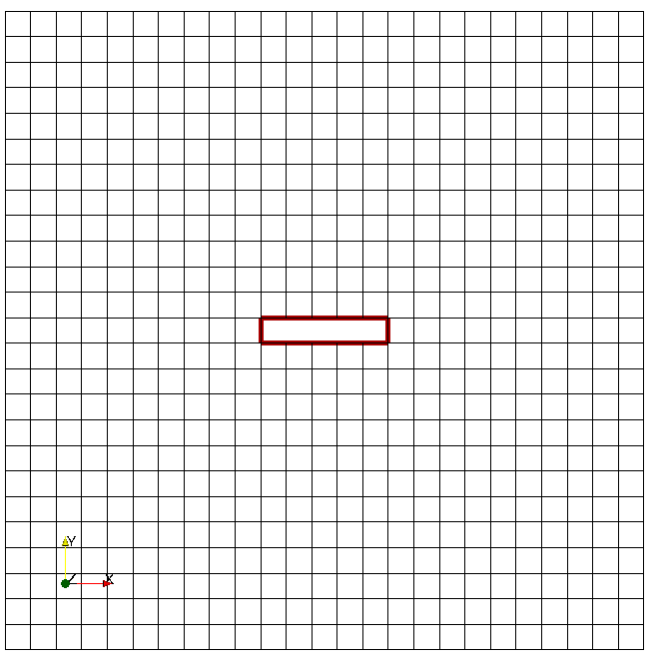
Using RBF in a CFD Code Context

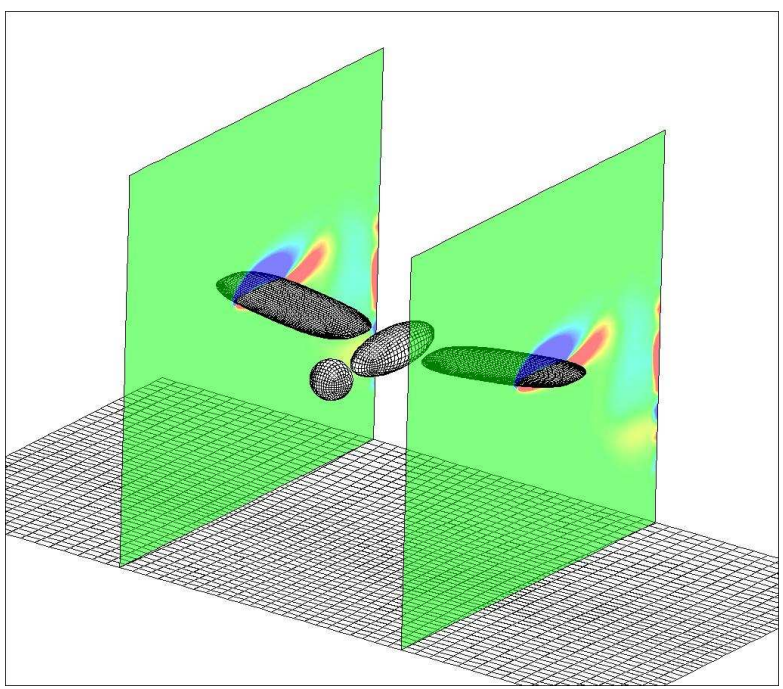
- Automatic mesh motion: mesh deformation based on a small number of control points located on moving boundaries: one-off interpolation cost
- RBF mesh morphing in geometric shape optimisation: morphing points control

Radial Basis Function

Radial Basis Function Automatic Mesh Motion

- Mathematical tool which allows data interpolation from a small set of control points to space **with smoothness criteria** built into the derivation
- Used for mesh motion in cases of large deformation: no inverted faces or cells
- Control points chosen on a moving surface, with “extinguishing function” used to control far-field mesh motion
- Implemented by Frank Bos, TU Delft and Dubravko Matijašević, FSB Zagreb





Radial Basis Function

RBF Mesh Morphing

- RBF morphing object defines the parametrisation of geometry (space):
  1. Control points in space, where the parametrised control motion is defined
  2. Static points in space, whose motion is blocked
  3. Range of motion at each control point: $(\mathbf{d}_0, \mathbf{d}_1)$
  4. Set of scalar parameters δ for control points, defining current motion as

$$\mathbf{d}(\delta) = \mathbf{d}_0 + \delta(\mathbf{d}_1 - \mathbf{d}_0), \quad \text{where} \quad 0 \leq \delta \leq 1$$

- For each set of δ parameters, mesh deformation is achieved by interpolating motion of control points $\mathbf{d}$ over all vertices of the mesh: new deformed state of the geometry
- Mesh in motion remains valid since RBF satisfies smoothness criteria

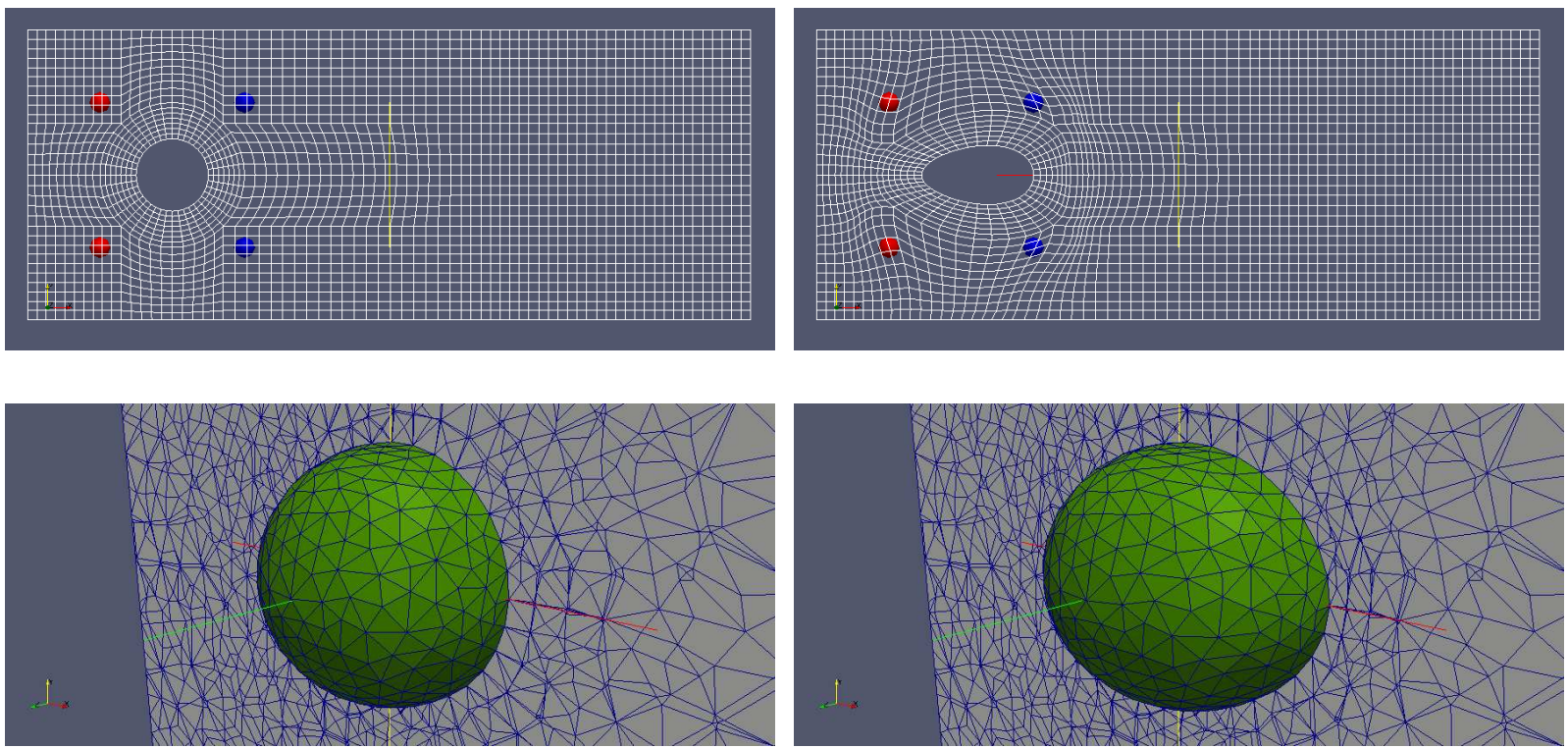
Using RBF in Optimisation

- Control points may be moved individually or share δ values: further reduction in dimension of parametrisation of space
- Mesh morphing state is defined in terms of δ parameters: to be controlled by the optimisation algorithm

Radial Basis Function

RBF Mesh Morphing: Cylinder and Sphere Examples

- Parametrisation uses a single parameter δ for this motion, with various number of control points

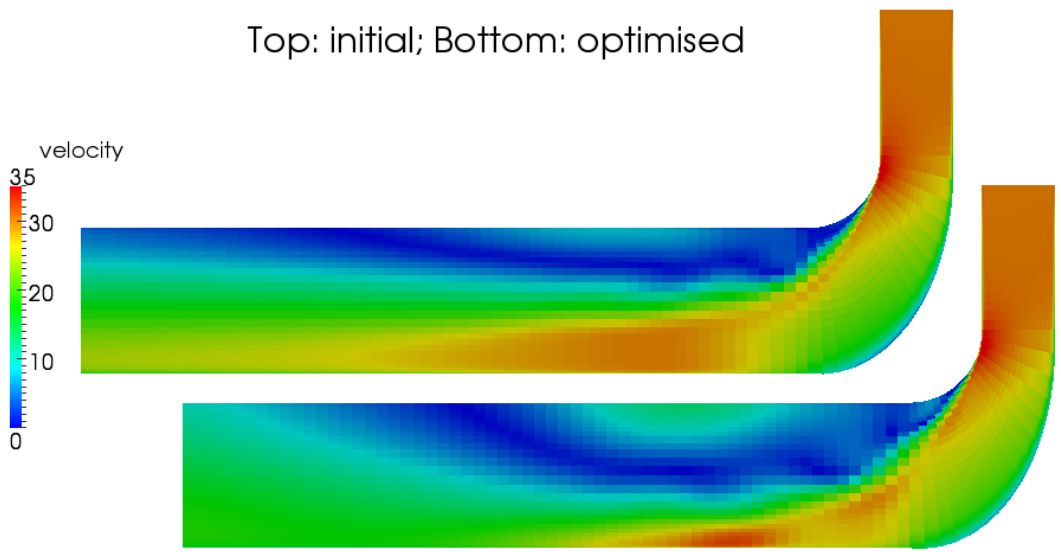


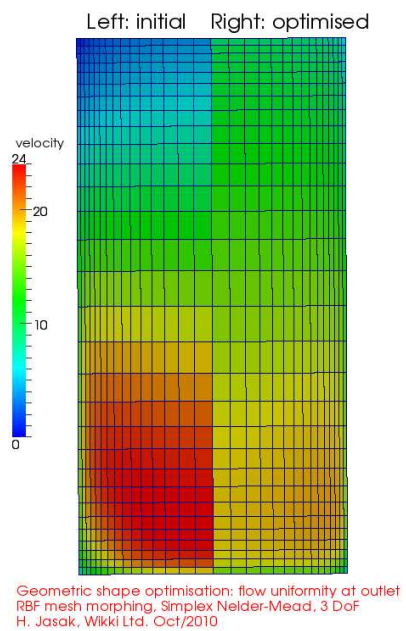
Geometric Shape Optimisation

HVAC 90 deg Bend: Flow Uniformity at Outlet

- Flow solver: incompressible steady-turbulent flow, RANS $k - \epsilon$ model; coarse mesh: 40 000 cells; 87 evaluations of objective with CFD restart
- RBF morphing: 3 control points in motion, symmetry constraints; 34 in total
- Objective: flow uniformity at outlet plane

```
iter = 0 pos = (0.9 0.1 0.1) v = 22.914 size = 0.69282
iter = 5 pos = (0.1 0.1 0.1) v = 23.0088 size = 0.584096
iter = 61 pos = ((0.990164 0.992598 0.996147) v = 13.5433 size = 0.00095
```





Non-gradient Based optimization

- Radial basis function and free form deformation are effective tools in geometrical parameterization
- By using the geometrical parameterization, the number of parameters is reduced from thousands to a manageable number (up to 25 parameters)
- Non-gradient algorithms of the simplex type are not effective when used with large number of parameters due to large deformations of the geometric simplex in parameter space and algorithms that avoid this problem must be used
- Gradient based optimization largely avoids these problems as it uses analytical and semi-analytical tools to compute the gradient and deduce the local behavior of the function

Gradient Based optimization

- Gradient based optimization algorithms rely on the analytical and semi-analytical tools of gradient evaluation in order to guide optimization algorithms
- OpenFOAM software framework allows a rapid implementation of PDE solvers
- If the original PDE equations are differentiated with respect to the optimization parameter, the resulting equations form a continuous approach to gradient evaluation
- Two approaches in analytical differentiation of PDEs and gradient evaluations:
  - Continuous tangent equations
  - Continuous adjoint equations

Continuous Analytical Gradient (1)

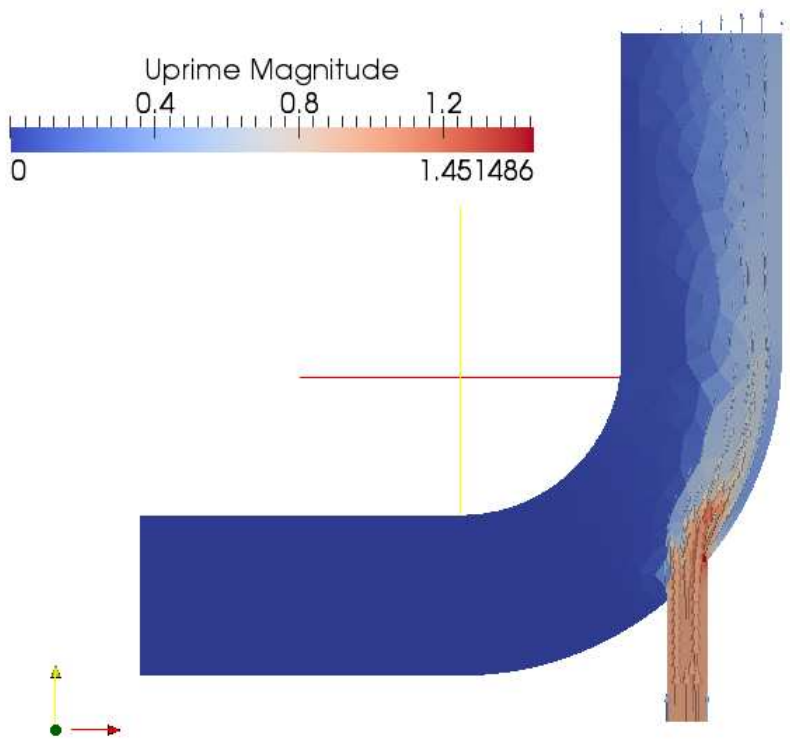
- Continuous tangent equations are obtained by performing the Getaux (Frechet) differentiation of the original PDEs
- In the case of the incompressible laminar Navier-Stokes equations, continuous tangent equations are as follows:

$$\begin{aligned} \nabla_{\bullet}\mathbf{u}' &= 0 \\ \mathbf{u}\nabla_{\bullet}\mathbf{u}' + \mathbf{u}'\nabla_{\bullet}\mathbf{u} - \nu\nabla_{\bullet}\nabla\mathbf{u}' &= -\nabla p' \end{aligned}$$

- This system of equations describes the propagation of the derivatives from inputs to outputs in the system

Analytical Gradient Based optimization

- An example of gradient of velocity field with respect to the inlet velocity in the smaller pipe computed by continuous tangent equations:



Continuous Analytical Gradient (2)

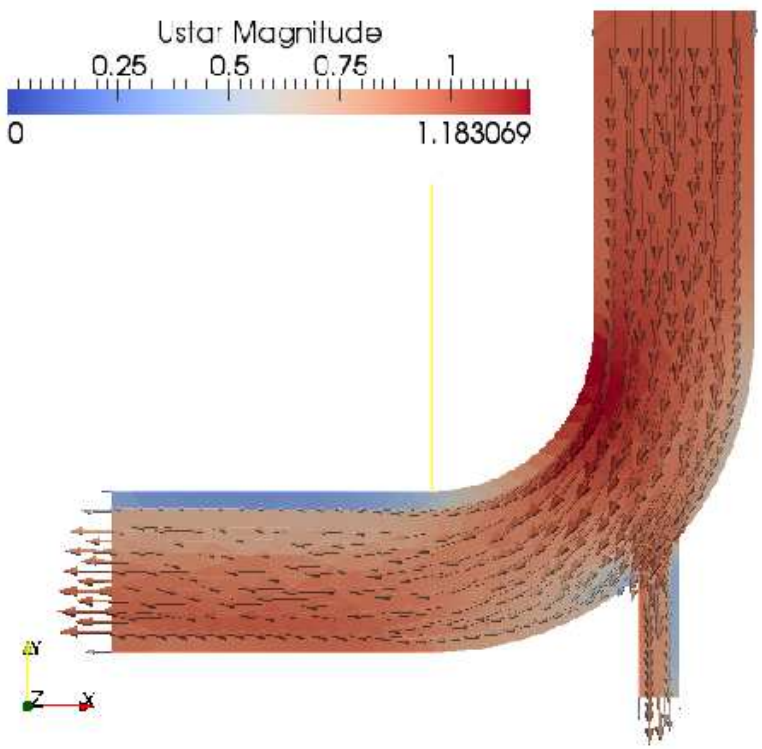
- Continuous adjoint equations are obtained by using the duality principles on continuous tangent equation
- In the case of the incompressible laminar Navier-Stokes equations, continuous adjoint equations are as follows:

$$\begin{aligned} -\nabla \bullet \mathbf{u}^* &= 0 \\ -\mathbf{u}\nabla \bullet \mathbf{u}^* - \nabla(\mathbf{u}^*) \bullet \mathbf{u} - \nu \nabla \bullet \nabla \mathbf{u}^* &= -\nabla p^* \end{aligned}$$

- This system of equations describes the propagation of the derivatives from outputs to inputs in the system

Analytical Gradient Based optimization

- An example of gradient of velocity field with respect to the outlet velocity in the larger pipe computed by continuous adjoint equations:



Discrete Analytical Gradient

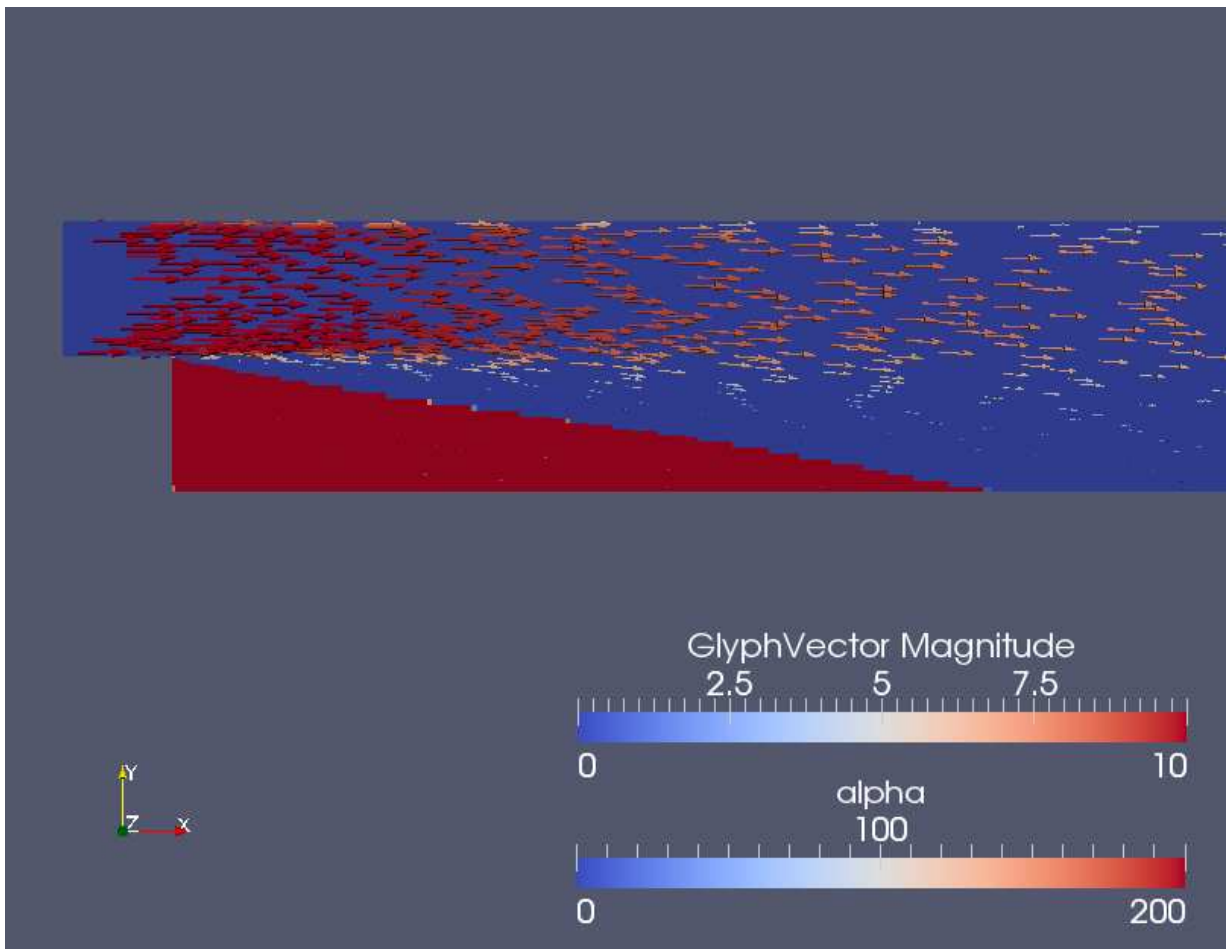
- Approaches in discrete analytical gradient evaluation:
  - By hand
  - Source code parsing
  - Operator overloading
- Currently in OpenFOAM there is operator overloading approach implemented in class FadOne
- Consistent use of this class leads to the discrete tangent code
- Using the templating of the code and operator overloading, the compiler insert a general code for derivative evaluation at the compilation time
- Gradients computed this way are accurate to the machine precision
- Discrete adjoint gradient evaluation is also possible based on this approach

Semi-Analytical Gradient

- The structure of the computational code can be reused in the so called semi-analytical approach to efficiently evaluate gradients
- This approach is similar in structure to discrete adjoint and tangent equations as the discrete structure of the code is re-used
- Derivatives are evaluated by adding a different forcing term to the right hand side of the discrete (implicit or explicit) system
- This approach is consistent with the original system of equations if exact gradients are used in the Newton linearization
- In OpenFOAM an implementation is possible by making minimal changes to the top level code

Topology Optimisation

- An example of a topology optimisation using porous media approach to block the flow and continuous adjoint equation for sensitivity:



Summary

- OpenFOAM is ideal for integration into an external optimisation loop
  - No license cost
  - Text file controlled
  - Integrated Pre- and Post-processing
  - Command-line driven workflows
- OpenFOAM software framework is ideal for advanced techniques (*e.g.* gradient based algorithms) due to existence of the necessary infrastructure for implementation of PDEs and algorithms
- Many tools for optimization already exist in OpenFOAM
- The best algorithm depends on the exact problem as well as the client's infrastructure and established workflows