Implementation of a continuous adjoint for topology optimization of ducted flows

Carsten Othmer*
Volkswagen AG, 38436 Wolfsburg, Germany

Eugene de Villiers†
Icon CG, London, W14 9DH, U.K.

Henry G. Weller‡
OpenCFD Ltd., Salfords, Surrey, RH1 5RG, U.K.

Topology optimization of fluid dynamical systems is still in its infancy, with its first academic realizations dating back to as late as four years ago. In this paper, we present an approach to fluid dynamic topology optimization that is based on a continuous adjoint. We briefly introduce the theory underlying the computation of topological sensitivity maps, discuss our implementation of this methodology into the professional CFD solver OpenFOAM and present first results obtained for the optimization of ducted flows wrt. dissipated power.

I. Fluid dynamic topology optimization

In structure mechanics, topology optimization is a well-established concept for design optimization with respect to tension or stiffness.[1] Its transfer to computational fluid dynamics, however, just began four years ago with the pioneering work of Borrvall and Petersson.[2] Since then, this topic has received quite some interest both in academia and in the industry[3−7].

The starting point for fluid dynamic topology optimization is a volume mesh of the entire installation space. Based on a computation of the flow solution inside this domain, a suitable *local* criterion is applied to decide whether a fluid cell is "good" or "bad" for the flow in terms of the chosen cost function. In order to iteratively remove the identified bad cells from the fluid domain, they are either punished via a momentum loss term, or holes are inserted into the flow domain, with their positions being determined from an evaluation of the so-called topological asymptotic.

In the former case, the momentum loss term is usually realized via a finite cell porosity, i. e. the whole design domain is treated as a porous medium: Each cell is assigned an individual porosity α_i, which is modeled via Darcy's law. The value of α_i determines if the cell is fluid-like (low porosity values) or has a rather solid character (high values of α_i). In other words, the porosity field controls the geometry, and the α_i are the actual design variables.

With such a setting, an adjoint method can be applied to elegantly compute the sensitivities of the chosen cost function wrt. the porosity of each cell. The obtained sensitivities can then be fed into a gradient-based optimization algorithm – possibly with some penalization of intermediate porosity values in order to enforce a "digital" porosity distribution, and after several iterations, the desired optimum topology is finally extracted as an isosurface of the obtained porosity distribution or simply as the collection of all non-porous cells.

In a recent study, Othmer et al.[7] were able to verify the applicability of this methodology to typical automotive objective functions, including dissipated power, equal mass flow through different outlets, flow uniformity and angular momentum of the flow in the outlet plane. In that proof-of-concept study, Automatic Differentiation techniques were applied to an academic CFD code in order to obtain a discrete adjoint solver. For industrial-sized problems, however, this code is not suitable. Therefore, we implemented the methodology via a continuous adjoint into the professional CFD environment OpenFOAM.[8] The underlying equations and

*Development Engineer, Numerical Analysis Dept.
†Consultant Engineer, Open Source Simulation Services.
‡Lead Programmer, OpenFOAM Development

their implementation will be presented in the following sections, before we present examples of topological sensitivity maps for an air duct segment.

II. Computation of topological sensitivity maps: theory

If J stands for the cost function to be minimized, the optimization problem can be stated as follows:

$$\text{minimize } J = J(\alpha, \mathbf{v}, p) \text{ subject to } \mathcal{R}(\alpha, \mathbf{v}, p) = 0\,, \tag{1}$$

where α represent the design variables, i. e. the porosity distribution, and $\mathbf{v}$ and p stand for velocity and pressure, respectively. $\mathcal{R} = (R_1, R_2, R_3, R_4)^T$ denotes the state equations, in our case the incompressible, steady-state Navier-Stokes equations:

$$(R_1, R_2, R_3)^T = (\mathbf{v}\cdot\nabla)\,\mathbf{v} + \nabla p - \nabla\cdot\Big(2\nu\mathsf{D}(\mathbf{v})\Big) + \alpha\mathbf{v} \tag{2}$$

$$R_4 = -\nabla\cdot\mathbf{v}\,, \tag{3}$$

with kinematic viscosity ν and the rate of strain tensor $\mathsf{D}(\mathbf{v}) = \nabla\mathbf{v} + (\nabla\mathbf{v})^T$. The essential component for the topology optimization methodology is the the Darcy term $\alpha\mathbf{v}$.

We thus have a constrained optimization problem, with the constraints being the state equations. Such problems are commonly tackled by introducing a Lagrange function L and reformulating the cost function as

$$L := J + \int_\Omega (\mathbf{u}, q)\,\mathcal{R}\,d\Omega\,, \tag{4}$$

where we have introduced the adjoint velocity $\mathbf{u}$ and the adjoint pressure q as Lagrange multipliers.

The first objective function of our implementation is the power dissipated by the fluid dynamic device. It can be computed as the net inward flux of energy, in our case total pressure, through the device boundaries:

$$J = -\int_\Gamma d\Gamma\,(p + v^2/2)\mathbf{v}\cdot\mathbf{n}\,. \tag{5}$$

This objective function involves only an integral over the outer surface of the flow domain and has no volume contribution from the domain itself. In such cases the adoint partial differential equations read:

$$-2\mathsf{D}(\mathbf{u})\,\mathbf{v} = -\nabla q + \nabla\cdot\Big(2\nu\mathsf{D}(\mathbf{u})\Big) - \alpha\mathbf{u} \tag{6}$$

$$\nabla\cdot\mathbf{u} = 0\,. \tag{7}$$

These are the equations that have to be implemented into the CFD solver, alongside with appropriate boundary conditions that depend on the chosen cost function. If $\mathbf{u}$ and q are then chosen to satisfy the adjoint equation system, the desired sensitivities can be computed according to Eqn. (4) as

$$\frac{\partial L}{\partial\alpha_i} = \frac{\partial J}{\partial\alpha_i} + \int_\Omega (\mathbf{u}, q)\,\frac{\partial\mathcal{R}}{\partial\alpha_i}\,d\Omega\,, \tag{8}$$

where $\partial L/\partial\alpha_i$ is the sensitivity of the cost function wrt. the porosity α_i of cell i. For the topology optimization methodology, the porosity is just an auxiliary variable to describe a continuous transition from fluid to solid. Therefore, there is no *explicit* dependence of the cost function on the porosity: $\partial J/\partial\alpha_i = 0$. Furthermore, as the porosity α_i enters the primal equation system only in cell i and only via the Darcy term, we can write

$$\frac{\partial\mathcal{R}}{\partial\alpha_i} = \begin{pmatrix}\mathbf{v}\\ 0\end{pmatrix}\chi_i \tag{9}$$

with χ_i being the characteristic function of cell i. Hence, according to Eqn. (8), we can finally compute the desired sensitivity for each cell as the scalar product of adjoint and primal velocity times the cell volume:

$$\frac{\partial L}{\partial\alpha_i} = \mathbf{u}_i\cdot\mathbf{v}_i\,V_i\,. \tag{10}$$

III. Solver implementation

OpenFOAM (Open **F**ield **O**peration **A**nd **M**anipulation) is a CFD toolbox that can be used to simulate a broad range of physical problems. The code was chosen as the development environment for the topology optimiser due partially to its open source (GNU General Public Licence – GPL) and therefore transparent nature, but primarily because of its high level symbolic application programming interface (API). The flexibility of this interface allows for a straight forward implementation of the continuous adjoint, using previously validated components that make up the other applications in the toolbox. Fig. 1 shows the source code for the adjoint solver component of the topology optimisation tool.

```
1    tmp<fvVectorMatrix> UEqn
2    (
3        fvm::div(-phi, U)
4      - fvc::grad(U) & V
5      + turbulence->divR(U)
6      + fvm::Sp(alpha, U)
7    );
8
9    solve(UEqn() == -fvc::grad(q));
10
11   volScalarField rAU = 1.0/UEqn().A();
12
13   U = rAU*UEqn().H();
14
15   phia = fvc::interpolate(U) & mesh.Sf();
16
17   fvScalarMatrix qEqn
18   (
19       fvm::laplacian(rAUa, q) == fvc::div(phia)
20   ); qEqn.solve();
21
22   phia -= qEqn.flux();
23
24   U -= rAU*fvc::grad(q);
```

Figure 1. Adjoint solver implementation in OpenFOAM.

The main symbols in Fig. 1 are defined as follows:

- phi - primal inter-cell volume flux
- phia - adjoint inter-cell volume flux
- q - adjoint pressure
- U - adjoint velocity
- V - primal velocity

where fluxes are calculated using the velocities from the previous iteration. Throughout "fvm::" prefaces implicit operators, while "fvc::" denotes the explicit equivalent. The solver uses a segregated approach and a SIMPLE-type algorithm to couple the adjoint velocities and pressure. Turbulence is assumed to be "frozen", so that the primal turbulent viscosity can be re-used for the adjoint diffusion term.

The adjoint momentum predictor is constructed in lines 1-7. It consists of the negative convection of the adjoint velocity by the primal (line 3). The next term (line 4), which describes the dot product of the adjoint gradient with the primal velocity, has to be represented explicitly, since the dot product of the gradient creates cross-coupling between the adjoint velocity components. The capability to solve cross-coupled equations is not currently available in the code library.

Turbulent and laminar diffusion is handled via a plug-in module, in this case denoted by the "turbulence" term which is previously defined. The final term in the momentum equation represents the effect of porosity

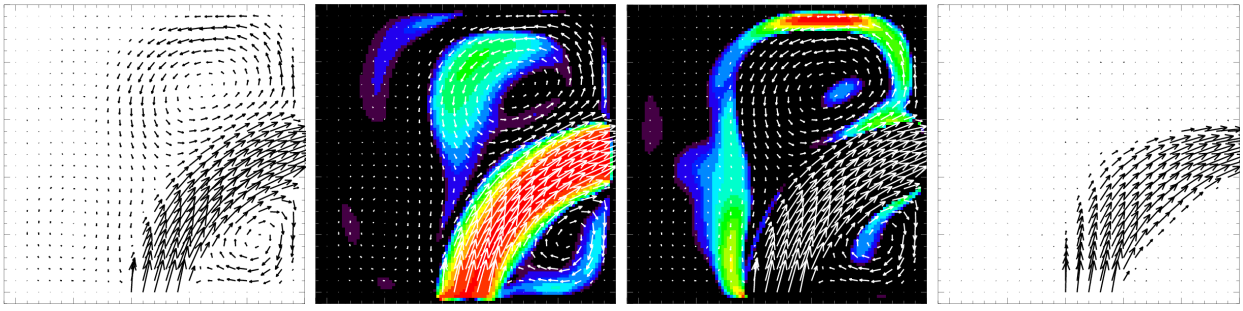
(*alpha*) used as a momentum sink in "bad" cells. The adjoint momentum is then solved by setting the matrix equal to the adjoint pressure gradient (line 9).

The construction of the adjoint pressure equation (lines 17-20) requires an intermediate adjoint flux found by dividing the right hand side of the Ueqn matrix (.H()) by the diagonal coefficient (.A()) (lines 11, 13 and 15). After the pressure has been solved the intermediate flux and velocity have to be updated so that they obey continuity (lines 22 and 24).

In addition to solving the adjoint equations, routines are also required to derive the sensitivity of the objective function to changes in porosity, to solve the primal equation system and to update the porosity. Each objective function also requires specialised boundary conditions that have to be derived from the governing equations. These additional steps will be described in the final paper.

IV. First applications

Fig. 2 illustrates the application of the developed adjoint code to a simple two-dimensional (2D) test case with 100×100 Cartesian grid cells. At a Reynolds number of 1000 the flow enters the box through an inlet on the South wall and leaves it through an outlet on the East wall. For the original flow field as shown in Fig. 2a, the sensitivities of the "good" cells (i. e. those with positive sensitivities) and the "bad" or counterproductive cells are displayed separately in Figs. 2b and c. As expected, the most important cells for the flow passage lie along the main path, whereas the counterproductive cells are mostly located in the backflow regions. Fig. 2d finally displays the flow field that was obtained after five unpenalized steepest descent iterations of porosity update. It constitutes a cost function improvement of 25% as compared to the original flow field.



(a) (b) (c) (d)

Figure 2. Application to a 2D example: (a) Initial velocity field, (b) sensitivities of "good" cells, (c) sensitivities of "bad" cells, (d) velocity field after five iterations.

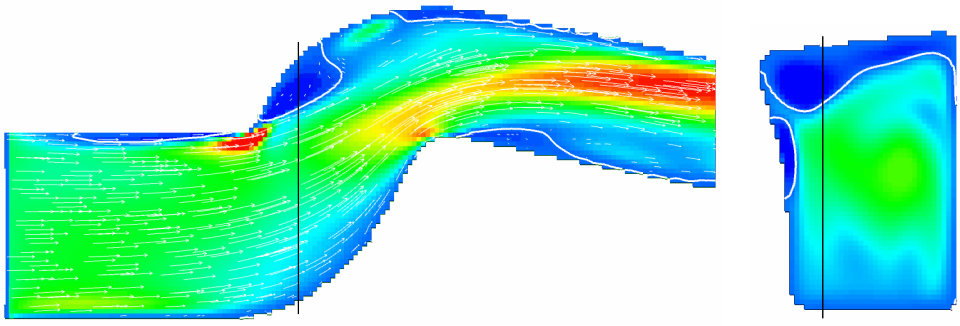


Figure 3. Sensitivities wrt. dissipated power for a segment of a Volkswagen air duct: The air enters the channel from the left hand side at a Reynolds number of 2500. Hot and cold colours correspond to good and bad cells, respectively. Isolines of zero sensitivity are shown in white.

As a 3D test case, we chose a segment of an air duct – a geometry that was well studied in previous works.[9] Fig. 3 shows the sensitivities wrt. dissipated power for this channel segment. "Hot" colours correspond to positive sensitivities, whereas the dark blue areas mark the counterproductive cells. The white contours are the isolines of zero sensitivities, i. e. the borderline between "good" and "bad" cells. As could be expected from physical reasoning, the counterproductive cells are located in the regions of detachment and backflow. How sensitivity maps of this kind can be used to generate optimal designs of industrially relevant, complex air duct geometries will be shown in the full paper.

V. Summary and Outlook

We have presented the theory underlying the computation of topological sensitivity maps and its implementation into the CFD environment OpenFOAM. While the full paper will contain examples of industrially relevant, complex geometries, our first applications of the code to a 2D test case and a 3D air duct segment already demonstrated the potential of the developed method.

Future work will be dedicated to an extension of the code towards other objective functions. For a large variety of objective functions this involves only adaptations of the boundary conditions. Thanks to the flexible high level symbolic application programming interface of OpenFOAM, such adaptations are straight forward within the developed adjoint solver framework.

References

[1]M.P. Bendsøe and O. Sigmund, Topology optimization: theory, methods, and applications, Springer, Berlin, 2004.

[2]T. Borrvall and J. Petersson, Topology optimization of fluids in Stokes flow, *Int. J. Num. Meth. Fluids*, **41**, p. 77, 2003.

[3]O. Sigmund, A. Gersborg-Hansen, and R.B. Haber, Topology optimization for multi-physics problems: A future FEMLAB application?, *Proc. Nordic MATLAB Conf. 2003*, L. Gregersen (Ed.), Comsol A/S, Søborg, Denmark, p. 237, 2003.

[4]L.H. Olesen, F. Okkels, and H. Bruus, Topology optimization of Navier-Stokes flow in microfluidics, *ECCOMAS 2004*, Jyväskylä, 2004.

[5]O. Moos, F.R. Klimetzek, and R. Rossmann, Bionic optimization of air-guiding systems, *SAE 2004-01-1377*, 2004.

[6]J.K. Guest and J.H. Prévost, Topology optimization of creeping flows using a Darcy-Stokes finite element, *Int. J. Num. Meth. Engng.*, **66** (3), p. 461, 2006.

[7]C. Othmer, Th. Kaminski, and R. Giering, Computation of topological sensitivities in fluid dynamics: Cost function versatility, *ECCOMAS CFD 2006*, Delft, 2006.

[8]http://www.opencfd.co.uk/openfoam/www.opencfd.org

[9]C. Othmer and Th. Grahs, Approaches to fluid dynamic optimization in the car development process, *EUROGEN 2005*, Munich, 2005.