Discrete Adjoint CHT Simulation in OpenFOAM

Markus Towara[1], Martin Spiller[2], Conrad Wächter[3]

17th OpenFOAM Workshop, Cambridge UK

[1] STCE, RWTH Aachen University
[2] IsaTEC Aachen
[3] Cloud & Heat Technologies GmbH, Dresden

Motivation

- STCE develops and maintains an AD (automatic differentiation) enabled fork of OpenFOAM
- Adjoint (reverse) and Tangent (forward) implementations available (+higher order)
- Adjoint AD gives accurate derivatives at (comparatively) cheap cost
- AD framework was applied to CHT problems in project *”Entwicklung optimierter Kühlgeometrien mittels adjungierter Simulationsmethoden für die Direkt-Heißwasserkühlung von Rechenzentren”*

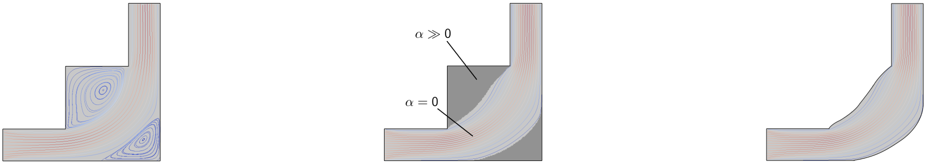
Motivation: Topology Optimization

- Augment NS momentum equations by source term $\mathbf{u}\alpha$:

$$(\mathbf{u} \otimes \boldsymbol{\nabla})\,\mathbf{u} = \nu \boldsymbol{\nabla}^2 \mathbf{u} - \frac{1}{\rho}\boldsymbol{\nabla} p - \alpha \mathbf{u}$$

- Parameter α allows to penalize cells of the geometry to redirect flow
- Penalty term can be interpreted as porosity, according to Darcy's law

$$\frac{\Delta p}{\Delta x} = -(\frac{\mu}{\kappa})\mathbf{u}$$



Initial design space

Optimized penalty field

Re-Parametrization

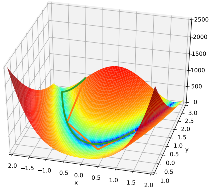
How to find appropriate penalty field α

- Define cost function $\mathcal{J}$, e.g. total power loss between inlet and outlet:

$$\mathcal{J} = -\int_{\Gamma} \left(p + \frac{1}{2} \|\mathbf{u}\|^2 \right) \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}\Gamma$$

- Calculate sensitivity of the cost function w.r.t. parameters α_i

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\alpha_i} = ???$$



- Calculate an updated porosity field α^{n+1}, e.g. using gradient descent:

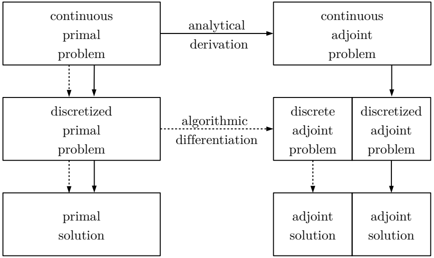
$$\alpha_i^{n+1} = \alpha_i^n - \lambda \cdot \frac{\mathrm{d}\mathcal{J}^n}{\mathrm{d}\alpha_i^n}, \quad \text{with constraints} \quad 0 \leq \alpha_i \leq \alpha_{\max}$$

- Loop until α converged...

Efficient optimization methods need gradients!

- Number of inputs to be optimized might be in the millions
- Calculating the gradient with finite differences (FD) extremely expensive
- Number of outputs usually $1 \leq m \ll n$
- Adjoint methods allow to calculate the gradient with only m additional (augmented) function evaluations

How to obtain adjoint sensitivities



Continuous method: (some adjoint solvers in OpenFOAM.com)

- *Differentiate first, discretize later*
- Derive adjoint equations analytically
- Implement, discretize, and solve adjoint equations along primal

+ Fast, physically interpretable

— Hard to derive, can be inconsistent to primal

Discrete method: (our fork)

- *Discretize first, differentiate later*
- Use implementation to get the derivatives (Algorithmic Differentiation)

+ Flexible, derivation automatic, sensitivities consistent to implementation

— Memory intensive, generally slower than continuous

AD Basics[1]

- Assume $y = f(x)$ with $x \in \mathbb{R}^n, y \in \mathbb{R}^m$
- Forward (tangent) AD: $\dot{y} = \dot{f}(x, \dot{x}) = \nabla f \cdot \dot{x}$

  Get Jacobian at cost $O(n \cdot cost(f))$ by letting $\dot{x} \in \mathbb{R}^n$ range over e_i
- Reverse (adjoint) AD: $\bar{x} = \bar{f}(x, \bar{y}) = \bar{y} \cdot \nabla f$

  Get Jacobian at cost $O(m \cdot cost(f))$ by letting $\bar{y} \in \mathbb{R}^m$ range over e_i
- Often $m \ll n$ or even $m = 1$ (e.g. scalar cost function)
- Modes can be recursively combined to obtain higher derivatives
- Sparsity in Jacobians / Hessians can be exploited by coloring approaches

[1] A. Griewank, A. Walther: Evaluating Derivatives, 2nd Edition

- Fork of `OpenFOAM.com`, currently based on OpenFOAM.com v2112
- We use operator overloading AD tool
- All floating point variables replaced by custom AD datatype (defined in `ad.hpp`)
- All libraries in `src/` are *"ADified"*
- Custom solvers build on top of base solvers
  (e.g. `simpleFoam` → `adjointSimpleFoam`)
- Currently no good interaction with existing vanilla OpenFOAM optimization capabilities :(

Implications and limitations of ADified OpenFOAM

- Comes at a cost due to operator overloading, memory allocation, reverse propagation
- Run-time increased by factor $\approx 5 - 20$
- Remember: All floating point variables replaced by custom AD datatype (no fully templated floating point datatype in OpenFOAM :( )
- $\Rightarrow$ can not (easily) fall back to double implementations for passive code
- Linear solver can be symbolically differentiated during reverse propagation

$$
\begin{aligned}
x = A \setminus b \Rightarrow \bar{b} &= A^T \setminus \bar{x} \\
\bar{A} &= -x \cdot \bar{b}^T
\end{aligned}
$$

- Parallelism by AdjointMPI (AMPI)

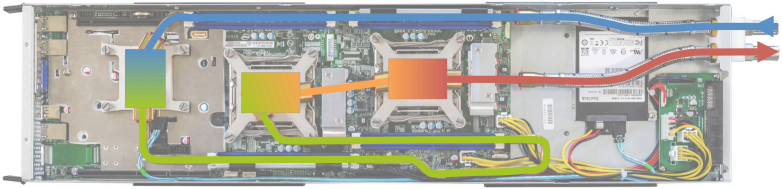
Features of Discrete Adjoint OpenFOAM

- Checkpointing full simulation runs (trading memory for run time)
- Reverse Accumulation (iterative re-evaluation of last iteration step)
- Piggy-Back optimization (design update with incomplete gradient)
- Implicit differentiation of residual:

$$\mathcal{R}(\alpha, x(\alpha)) = 0 \Leftrightarrow \frac{\partial \mathcal{R}}{\partial x}\frac{\partial x}{\partial \alpha} = -\frac{\partial \mathcal{R}}{\partial \alpha}$$

  should be exploited more, can also use tangent mode or FD due to sparsity.

- Cloud & Heat builds custom server solutions with water cooling
- Wants to re-use recovered heat instead of just convecting away to outside air
- Has to work with high inlet temperatures, want to maximize temperature delta



Project partner contributions

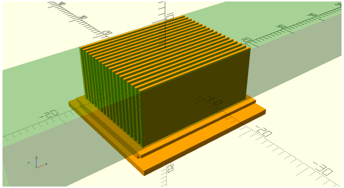
- STCE developed adjoint optimization technology in OpenFOAM
- IsaTEC performed validation studies with FloEFD and investigated fitting of the metal cooler geometries into plastic flow channels
- Cloud & Heat designed and constructed test chamber and ran tests

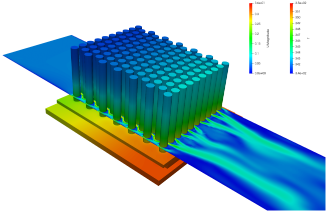
Challenges of Adjoint Conjugate Heat Transfer (CHT)

- (One) goal: Have cost function evaluated in different region than independents
  - E.g.: Cost function: Average die temperature
  - E.g.: Independents: Node positions of fluid-to-solid boundary nodes
- Source of complexity: fluid and solid regions with implicit coupling
  - slow convergence of temperature
- Meshes on both sides not necessary conforming on the interface
  - need to differentiate through point to point interpolation on interface
- Boundary conditions used have *hidden* data members which have to be explicitly handled with our checkpointing approach
- For Topology optimization: Physical properties of solid domain have to be replicated in fluid regions where artificial material is placed.

Simulations performed

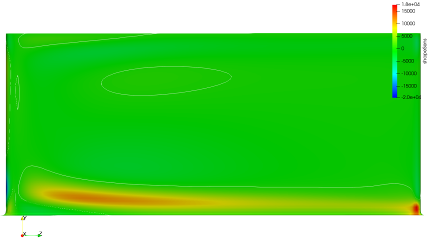
- Parametric simulation (identify suitable baseline):
  - Geometry generation with OpenSCAD
  - Pre-processing (splitting into sub-stls) w. python
  - Meshing with cfMesh
  - Simulation with chtMultiRegionSimpleFoam
  - Glued together with bash
  - Studied different configurations of fin and pin coolers

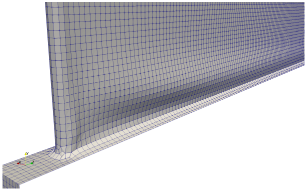




Adjoint Simulation

- Adjoint simulation:
  - Simulate single fin with symmetries (to reduce memory requirements)
  - Treat surface mesh points as parameters
  - Morph mesh in direction of normals, scaled by sensitivities





Summary:

- AD enables efficient and accurate computation of derivatives
- Disrcete adjoint OpenFOAM applied to heat transfer problems

Want to apply AD to your own problems?

- Discrete adjoint OpenFOAM is available as open source on request
- `https://stce.rwth-aachen.de/foam`

Simulations were performed with computing resources granted by RWTH Aachen University under project rwth0442.

Project supported by the Federal Ministry for Economical Affairs and Enegy (BMWi), on the basis of a decision by the German Bundestag