Topological Design Based on Highly Efficient Adoints Generated by Automatic Differentiation

Thomas Kaminski\textsuperscript{1}, Ralf Giering\textsuperscript{1}, and Carsten Othmer\textsuperscript{2}

\textsuperscript{1}FastOpt (http://FastOpt.com)
\textsuperscript{2}Volkswagen AG
CFD optimisation with sensitivity maps: Examples

Shape Optimisation
- Drag
- Pressure Drop

Topology Optimisation
- Dissipated Power
- Uniformity
Why topology optimisation?

Typical air duct geometries for cabin ventilation:

• Have to fit within very complicated design domain
• Challenging parametrization:
  Maximum design freedom vs. controllability of domain restrictions
Topology optimization with adjoint methods

So far, no professional tool available for CFD topology optimisation.

Approach pursued at VW:

- Treat design domain as porous medium
- **Continuous** transition between fluid and solid: Porosity $\alpha_{ijk}$
- Nominate target quantity $y$, e.g. dissipated energy
- Use **adjoint** to compute sensitivities, i.e. $\partial y / \partial \alpha_{ijk}$
- Update porosity accordingly
- Iterate towards a “digital“ porosity distribution
- Optimal topology = non-porous cells
Sensitivities via AD

Statements in code define elementary functions such as +, /, **, sin, exp...

Numerical Model

- $f_1$
- $f_2$
- $\ldots$
- $f_{N-1}$
- $f_N$

vector of parameters, e.g. design variables here: porosity

target quantity, e.g. lift, drag... here: dissipated energy

FastOpt
Sensitivities via AD: Tangent

Cost of gradient evaluation proportional to # of parameters:
One run of TLM per gradient component

Derivatives of elementary functions are simple,
they define local Jacobians

Tangent Linear Model

\begin{align*}
\delta \tilde{m} & \quad \rightarrow \quad Df_1 \quad \rightarrow \quad Df_2 \quad \rightarrow \quad \cdots \quad \rightarrow \quad Df_{N-1} \quad \rightarrow \quad Df_N \\
& \quad \text{Applies chain rule in forward direction} \\
\n\n& \quad \nabla y \cdot \delta \tilde{m}
\end{align*}
Sensitivities via AD: Adjoint

Cost of gradient evaluation **independent** of # of parameters:
One run of adjoint for **entire** gradient
But: Reversal of control flow complicates coding

\[ \nabla y = \delta y = 1 \]

Applies chain rule in reverse direction

*Adjoint Model*
AD of Solver

3D Solver based on NAST2D (Griebel, 1998)

From the Fortran 90 source code of the primal solver, AD tool TAF has generated:

• Tangent linear code

• Adjoint code for steady problems (Giering et al., FGCS, 2005) linearises around converged primal flow

• Adjoint code for general problems suitable for time dependent or not fully converged problems stores sequence (history) of primal flows

Performance on Linux Pentium 1.86 GHz for $15^3$ grid cell test configuration

<table>
<thead>
<tr>
<th></th>
<th># of lines w/o comments</th>
<th>CPU (solve+grad)</th>
<th>rel acc. vs. FD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal</td>
<td>2700</td>
<td>1.0</td>
<td>1E-8</td>
</tr>
<tr>
<td>TLM</td>
<td>3300</td>
<td>1.3</td>
<td>1E-8</td>
</tr>
<tr>
<td>ADM steady</td>
<td>3700</td>
<td>1.8</td>
<td>1E-5</td>
</tr>
<tr>
<td>ADM general</td>
<td>3700</td>
<td>1.8</td>
<td>1E-8</td>
</tr>
</tbody>
</table>
Automated Procedure

- Slight initial code modifications of primal solver code were necessary, to allow use in an optimisation loop and assure compliance with TAF.
- The generated code was not modified.
- The automation allowed for quick updates of the derivative code to changes of the primal solver.
- Used 4 significantly different formulations of the target function: dissip. energy, flow uniformity, equal mass, and tumble.
- After each update the derivative code is verified against the derivative approximation by finite differences of primal solves:
Adjoint approach: 2D laminar test case

Velocity (Re=2500)

E=1.0
Adjoint approach: 2D laminar test case

Sensitivities: “good“ cells

E=1.0
Adjoint approach: 2D laminar test case

Sensitivities: “bad“ cells

E=1.0
Adjoint approach: 2D laminar test case

Porosity (iter.1)

E=0.88
Adjoint approach: 2D laminar test case

Porosity (iter.2)

E=0.79
Adjoint approach: 2D laminar test case

Porosity (iter.3)

E=0.75
Adjoint approach: 2D laminar test case

Porosity (iter.4)

E=0.74
Adjoint approach: 2D laminar test case

Final velocity

E=0.74
Sensitivity and optimum depend on target function

- Sensitivities for air duct example: good and bad cells

Dissipated power

Flow uniformity
Combined target function

- Sensitivities for air duct example: good and bad cells
- $1 \times \text{uniformity} + 2 \times \text{dissipated power}$
Conclusions

+ Method fits ideally into design process
+ Delivers an unbiased design from scratch
+ Design domain restrictions are fulfilled automatically
+ Only tool to generate baffles etc. within the flow domain
- Less accurate flow solution (stepped geometry)

→Very powerful method for drafting duct geometries

AD via TAF ...

• has enabled the “proof of principle“
• allowed to demonstrate the variability of possible target functions
## Further Derivative Codes Generated by TAF

<table>
<thead>
<tr>
<th>Model (Who)</th>
<th>Area</th>
<th>Lines</th>
<th>Lang</th>
<th>TLM</th>
<th>ADM</th>
<th>Ckp</th>
<th>HES</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA/GMAO (w. Todling et al.)</td>
<td>Atmos</td>
<td>87,000</td>
<td>F90</td>
<td>1.5</td>
<td>4–11</td>
<td>2 lev</td>
<td>-</td>
</tr>
<tr>
<td>MOM3 (Galanti &amp; Tziperman)</td>
<td>Ocean</td>
<td>50,000</td>
<td>F77</td>
<td>Yes</td>
<td>4.6</td>
<td>2 lev</td>
<td>-</td>
</tr>
<tr>
<td>MITGCM (ECCO Consortium)</td>
<td>Ocean</td>
<td>170,000</td>
<td>F77</td>
<td>1.8</td>
<td>5.5</td>
<td>3 lev</td>
<td>11.0/1</td>
</tr>
<tr>
<td>BETHY (w. Knorr, Rayner, Scholze)</td>
<td>Land</td>
<td>5,400</td>
<td>F90</td>
<td>1.5</td>
<td>3.6</td>
<td>2 lev</td>
<td>12.5/5</td>
</tr>
<tr>
<td>Nav.-Stokes-Solver (Hinze, Slawig)</td>
<td>Aero</td>
<td>450</td>
<td>F77</td>
<td>-</td>
<td>2.0</td>
<td>steady</td>
<td>-</td>
</tr>
<tr>
<td>NSC2KE (w. Slawig)</td>
<td>Aero</td>
<td>500</td>
<td>F77</td>
<td>2.4</td>
<td>3.4</td>
<td>steady</td>
<td>9.8/1</td>
</tr>
<tr>
<td>HB_AIRFOIL (Thomas &amp; Hall)</td>
<td>Aero</td>
<td>8,000</td>
<td>F90</td>
<td>-</td>
<td>3.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ARPS (Yang, Xue, Martin) in progress</td>
<td>Atmos</td>
<td>40,000</td>
<td>F90</td>
<td>2.0</td>
<td>11.0</td>
<td>2 lev</td>
<td>-</td>
</tr>
<tr>
<td>NIRE-CTM</td>
<td>Atmos</td>
<td>860</td>
<td>F77</td>
<td>1.0</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOWer (MEGADESIGN) in progress</td>
<td>Aero</td>
<td>160,000</td>
<td>F77</td>
<td>3.3</td>
<td>6–10</td>
<td>steady</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Lines**: total number of Fortran lines without comments
- **Numbers for TLM and ADM give CPU time for (function + gradient) relative to forward model**
- **HES format**: CPU time for Hessian * n vectors rel. t. forw. model/ n
- **2 (3) level checkpointing costs 1 (2) additional model run(s)**
Derivatives of C Codes generated by TAC++

- TAC++ : AD for C(++) codes
- Transfer of TAF concepts/algorithms
- Still less mature than TAF
- But: fully automated generation of some smaller codes:

<table>
<thead>
<tr>
<th>Model (Coworkers)</th>
<th>Area</th>
<th>#lines of code</th>
<th>FUNC [s]</th>
<th>TLM / FUNC</th>
<th>ADM/ FUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roeflux (Cusdin, Mueller)</td>
<td>Aero</td>
<td>140</td>
<td>6.2E-7</td>
<td>3.3</td>
<td>3.9</td>
</tr>
<tr>
<td>2streams (Pinty et al.)</td>
<td>Satellite</td>
<td>~330</td>
<td>3.9E-6</td>
<td>2.0</td>
<td>4.4</td>
</tr>
<tr>
<td>TAU-ij (Gauger et al.)</td>
<td>Aero</td>
<td>~130</td>
<td>3.0E-3</td>
<td>--</td>
<td>2.6</td>
</tr>
<tr>
<td>LIBOR (Giles, Glasserman)</td>
<td>Finance</td>
<td>~210</td>
<td>2.0E-1</td>
<td>1.5</td>
<td>3.3</td>
</tr>
<tr>
<td>GasNetOpt (Steinbach)</td>
<td>Network</td>
<td>25</td>
<td>2.4E-7</td>
<td>2.4</td>
<td>2.7</td>
</tr>
<tr>
<td>Roeflux; F77, TAF</td>
<td></td>
<td>105</td>
<td>5.1E-7</td>
<td>--</td>
<td>2.9</td>
</tr>
</tbody>
</table>
AD of FLOWer within MEGADESIGN

Joint work with DLR (Eisfeld, Gauger, Kroll, Raddatz) in progress ...

<table>
<thead>
<tr>
<th>Method</th>
<th># code lines</th>
<th>Memory</th>
<th>CPU (solve+grad)</th>
<th>rel acc. vs FD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal</td>
<td>166,000</td>
<td>1</td>
<td>1.0</td>
<td>1E-08</td>
</tr>
<tr>
<td>TLM</td>
<td>268,000</td>
<td>~2</td>
<td>~3</td>
<td>1E-08</td>
</tr>
<tr>
<td>ADM steady</td>
<td>310,000</td>
<td>2—3</td>
<td>6—10</td>
<td>1E-05</td>
</tr>
<tr>
<td>ADM general</td>
<td>310,000</td>
<td>variable</td>
<td>&lt;10</td>
<td>1E-08</td>
</tr>
</tbody>
</table>

Sensitivity by FLOWer adjoint
NACA12, single grid, Wilcox Turbulence