Applications of differentiated CAD kernel in industrial shape optimisation

Mladen Banovic, Orest Mykhaskiv, Salvatore Auriemma
Andrea Walther, Jens-Dominik Müller, Herve Legrand

mladen.banovic@math.uni-paderborn.de

16th February, 2017
AD workshop, INRIA, France
CAD-based shape optimisation

Given a CAD-model with design parameters $\alpha$

Gradient-based optimisation

$$ \min_{\alpha} J(U(X_s(\alpha)), X_s(\alpha), \alpha) $$  \hspace{1cm} (1)

$$ R(U(X_s(\alpha)), X_s(\alpha)) = 0 $$  \hspace{1cm} (2)

$$ \frac{dJ}{d\alpha} = \frac{dJ}{dX_S} \frac{dX_S}{d\alpha} $$  \hspace{1cm} (3)

- CFD sensitivity $\frac{dJ}{dX_S}$: Adjoint method (efficiency)
- CAD sensitivity $\frac{dX_S}{d\alpha}$: Forward and Reverse Automatic Differentiation
OpenCASCADE Technology

OpenCASCADE Technology (OCCT) is an open source C++ library, consisting of thousands of classes and providing solutions in the areas of:

- Surface and solid modelling: to model any type of object,
- 3D and 2D visualization: to display and animate objects,
- Data exchange (import and export standard CAD formats) and tree-like data model.
Test-case 1: U-bend
Test-case 2: TU Berlin TurboLab Stator
Calculating the CAD sensitivities

In order to calculate the shape derivatives w.r.t. design parameters, **OCCT** has been **differentiated** by integrating the AD tool **ADOL-C**.

**Automatic Differentiation by OverLoading in C++**

ADOL-C uses **operator overloading** concept to compute first and higher derivatives of vector functions that are written in C or C++.

<table>
<thead>
<tr>
<th>Options</th>
<th>Differentiation modes</th>
<th>Integrated to OCCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>trace-based</td>
<td>forward, reverse forward</td>
<td>In progress</td>
</tr>
<tr>
<td>traceless</td>
<td>forward</td>
<td>✓</td>
</tr>
</tbody>
</table>

Mladen Banovic 5 / 23 16.02.2017
CAD sensitivities on U-bend
Run-time ratios - Traceless forward mode

- 1 + 1.5p
- 1 + 1p

```
g++ v4.8.5
clang++ v3.7.0
```

```
Number of directions (p)
0  20  40  60  80  100  120  140  160
Run-time ratio
0   20   40   60   80  100  120  140  160
```
Memory req. - Traceless forward mode

![Graph showing total memory consumption vs. number of directions]
Optimisation example that validates the reverse mode differentiation of OCCT

1. Construct two U-bends: original and perturbed one.
2. Sample final B-Spline surfaces with 12K pairs of \((u, v)\) parametric coordinates.
3. Define a cost function as the sum of squared distances of all \((x, y, z)\) point pairs.
4. Declare the original design parameters as independent variables of the system.
5. Minimise the cost function by using the L-BFGS-B optimisation algorithm.
6. Run the whole optimisation twice to compare two versions of differentiated OCCT kernel.
Optimisation example that validates the reverse mode differentiation of OCCT
**Results - Traceless forward vs Reverse mode of AD**

![Graph showing cost-function value vs number of iterations]

- **Cost-function value**
  - Traceless Forward mode
  - Trace-based Reverse mode
  - Abs. difference

![Graph showing difference vs gradient index]

- **Differentiation mode**
  - Traceless forward
  - Reverse

<table>
<thead>
<tr>
<th>Differentiation mode</th>
<th>Average duration (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traceless forward</td>
<td>13.7</td>
</tr>
<tr>
<td>Reverse</td>
<td>6.2</td>
</tr>
</tbody>
</table>

**Improved efficiency with the reverse mode: 55%**
Complete design chain

STAMPS solver

CFD Solver from QMUL
Discrete Adjoint AD
Spring/Elasticity Analogy
Mesh Perturbation

OCCT AD

Build Shapes ($\alpha$)
Find Mesh on CAD
Parametric Mesh $(u, v)$
Sensitivities in $(u, v)$

Optimiser

Steepest Descent

$$\frac{dJ}{d\alpha} = \frac{dJ}{dX_S} \frac{dX_S}{d\alpha}$$

$$\alpha^{(n+1)} = A(\alpha^{(n)}, \frac{dJ}{d\alpha}(\alpha^{(n)}))$$
Evaluating the total gradient $dJ/d\alpha$

1. Compute the $CAD$ sensitivity independently by using the OCCT kernel differentiated in the forward $traceless$ mode and couple it with the $CFD$ sensitivity.

2. Use the reverse mode of AD inside OCCT, thus having a full reverse mode differentiation of the complete differentiated design chain.
vector<adouble> designParameters = ...; //init the values
//put the derivative seeds
for(int i = 0; i < nParams; ++i)
    designParameters[i].setADValue(i, 1.);
//construct the CAD model
Handle(Geom_BSplineSurface) finalSurf = ConstructGeometry(designParameters);
//calculate the surface points derivatives w.r.t. design parameters
vector<double> cadSens;
for(double u = 0.; u <= 1.; u+=step)
{
    for(double v = 0.; v <= 1.; v+=step)
    {
        gp_Pnt aPnt;
        finalSurf->D0(u, v, aPnt);
        for(int i = 0; i < nParams; ++i)
        {
            cadSens.push_back(aPnt.X().getADValue(i));
            cadSens.push_back(aPnt.Y().getADValue(i));
            cadSens.push_back(aPnt.Z().getADValue(i));
        }
    }
} // ... cadSens x cfdSens ... => total gradient
Evaluating the total gradient

Reverse mode using ADOL-C trace (I)

Generating the trace

```cpp
vector<double> designParameters = ...; //init the values
adouble* adParameters = new adouble[nParams];
//start tracing, set the independents
trace_on(1);
for(int i = 0; i < nParams; ++i)
    adParameters[i] <<= designParameters[i];
//construct the CAD model and evaluate the surface points
Handle(Geom_BSplineSurface) finalSurf = ConstructGeometry(designParameters);
for(double u = 0.; u <= 1.; u+=step)
{
    for(double v = 0.; v <= 1.; v+=step)
    {
        gp_Pnt aPnt;
        finalSurf->D0(u, v, aPnt);
        aPnt.X() >>= output; //marking the dependents
        aPnt.Y() >>= output; //marking the dependents
        aPnt.Z() >>= output; //marking the dependents
    }
}
//end tracing
trace_off();
```
Reverse mode using ADOL-C trace (II)

Evaluating trace - calculating the gradient

double *independent = new double[nParams];

for (int i = 0; i < nParams; ++i)
    independent[i] = designParameters[i];

double *cfdSens = GetCfdSensitivities1dArray();

double *gradient = new double[nParams];

// use the vector jacobian product to evaluate the total gradient
vec_jac(1, sizeOfCfdSens, nParams, 0, independent, cfdSens, gradient);
Parametric U-bend optimisation results with the STAMPS flow solver (I)
Parametric U-bend optimisation results with the STAMPS flow solver (II)
Parametric U-bend optimisation results with the STAMPS flow solver (III)

Reduction of the total pressure loss = 17.9%.
TUB Stator optimisation results with the STAMPS flow solver (I)
TUB Stator optimisation results with the STAMPS flow solver (II)

![Graph of Total Pressure, Pa over a range from 0 to 20 with a 18% decrease]
Conclusions

Current status
✓ Differentiated full CAD-system
✓ Coupled with adjoint CFD - complete differentiated design chain.
✓ Reverse mode differentiation of OCCT validated for U-bend.

Further work
▶ Validate reverse mode differentiation of OCCT on TU Berlin TurboLab Stator.
Research is conducted within IODA\textsuperscript{1} project

\textit{Industrial Optimal Design using Adjoint CFD}

\textsuperscript{1}http://ioda.sems.qmul.ac.uk/ Grant Agreement No. 642959.