AD-based Discrete Adjoints in SU2

Tim Albring, Nicolas R. Gauger, Max Sagebaum, Beckett Y. Zhou

Chair for Scientific Computing

University of Kaiserslautern

67663 Kaiserslautern, Germany

nicolas.gauger@scicomp.uni-kl.de

1st SU2 Developers' Meeting, September 5-6, 2016, TU Delft

Chair for Scientific Computing

Head:

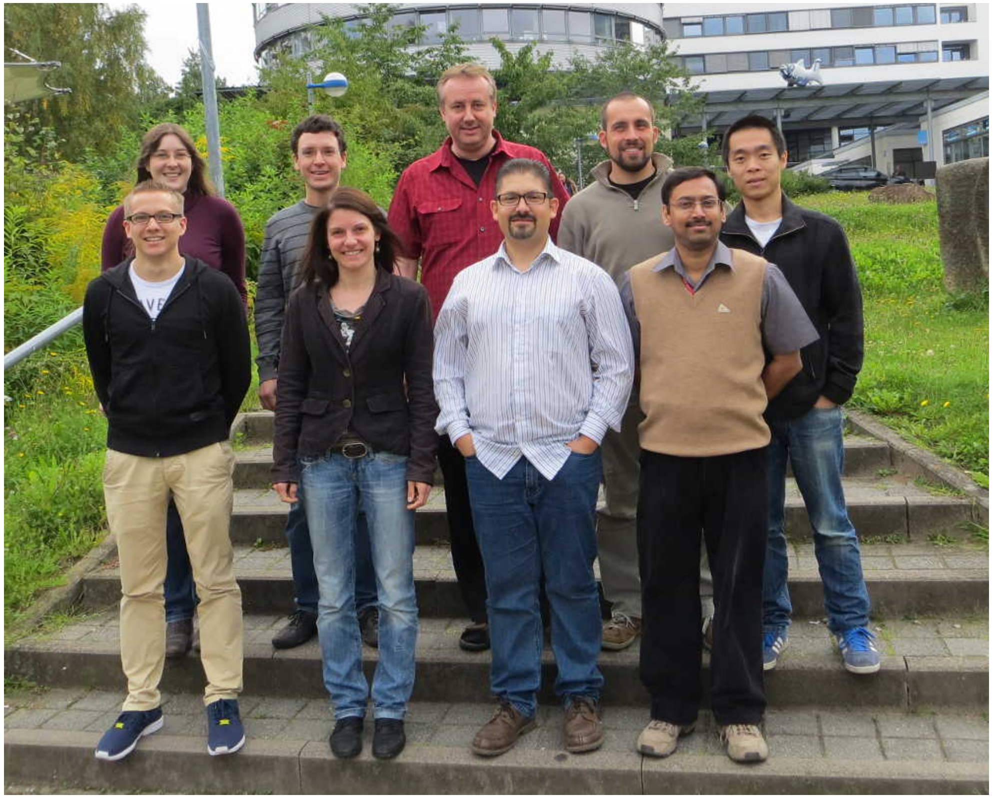
Prof. Dr. Nicolas R. Gauger

Secretary:

Ulrike Hahn

Researchers:

Tim Albring
Beckett Y. Zhou
Junis Abdel Hay
PD Dr. Josef Schüle
Dr. Emre Özkaya
Max Sagebaum
Lisa Kusch
Matthias Sonntag
Stefanie Günther



Caslav Ilic (DLR)
Mohammed Abu-Zurayk (DLR)
Sebastian Mann (MTU Aero Engines)
Mathias Luers (MTU Aero Engines)
Maarten Blommaert (FZ Jülich)

Chair for Scientific Computing

*) SU2 / CoDiPack

Head:

Prof. Dr. Nicolas R. Gauger *)

Secretary:

Ulrike Hahn

Researchers:

Tim Albring *)
Beckett Y. Zhou *)
Junis Abdel Hay
PD Dr. Josef Schüle
Dr. Emre Özkaya
Max Sagebaum *)
Lisa Kusch
Matthias Sonntag
Stefanie Günther

Caslav Ilic (DLR)
Mohammed Abu-Zurayk (DLR)
Sebastian Mann (MTU Aero Engines)
Mathias Luers (MTU Aero Engines)
Maarten Blommaert (FZ Jülich)

Outline & *Further Activities*

- **AD-based Discrete Adjoints**
- **Implementation in SU2**
- **Applications and Performance**
- **Coupled CFD-FWH Noise Prediction and Optimization Framework**

  (joint with Tom Economon and Carlos Ilario da Silva, Stanford)
- **AD-based Optimization of Multi-Stage Turbomachines**

  (joint with Salvatore Vitale and Matteo Pini, TU Delft)

- ***Ongoing Work on Fluid-Structure Adjoints***

  (joint with Ruben Sanchez Fernandez and Rafael Palacios, Imperial College)

Optimality System

- **Optimization Problem:**

$$\min_{\phi \in \Phi} J(W,\phi) \quad s.t. \quad R(W,\phi) = 0$$

- **Lagrangian:**

$$L = J + \Lambda^T R$$

- **Optimality condition (KKT system, 1. order necessary cond.):**

$$\frac{\partial L}{\partial \Lambda} = R \overset{!}{=} 0$$ State equation

$$\frac{\partial L}{\partial W} = \frac{\partial J}{\partial W} + \Lambda^T \frac{\partial R}{\partial W} \overset{!}{=} 0$$ Adjoint state equation

$$\frac{\partial L}{\partial \phi} = \frac{\partial J}{\partial \phi} + \Lambda^T \frac{\partial R}{\partial \phi} \overset{!}{=} 0$$ Design equation

Optimality System

- **Optimization Problem:**

$$\min_{\phi \in \Phi} J(W,\phi) \quad s.t. \quad R(W,\phi) = 0$$

- **Lagrangian: instead** $L = J + \Lambda^T R$**, continuous** L**:**

$$L = J + \left\langle \Lambda, R \right\rangle_{H^*,H}$$

- **Optimality condition (KKT system, 1. order necessary cond.):**

$$\frac{\partial L}{\partial \Lambda} = R \overset{!}{=} 0$$ State equation

$$\frac{\partial L}{\partial W} = \frac{\partial J}{\partial W} + \Lambda^T \frac{\partial R}{\partial W} \overset{!}{=} 0$$ Adjoint state equation

$$\frac{\partial L}{\partial \phi} = \frac{\partial J}{\partial \phi} + \Lambda^T \frac{\partial R}{\partial \phi} \overset{!}{=} 0$$ Design equation

Optimality System

- **Optimization Problem:**

$$\min_{\phi\in\Phi} J(W,\phi) \quad s.t. \quad R(W,\phi)=0 \quad \leftrightarrow$$

Fixed point iteration:

$$W = G(W,\phi)$$

- **Lagrangian: instead** $L = J + \Lambda^T R$**, continuous or discrete** L**:**

$$L = J + \langle \Lambda, R \rangle_{H^*,H} \quad \leftrightarrow \quad L(W,\Lambda,\phi) = J(W,\phi) + \Lambda^T (G(W,\phi) - W)$$

- **Optimality condition (KKT system, 1. order necessary cond.):**

$$\frac{\partial L}{\partial \Lambda} = R \overset{!}{=} 0$$

State equation

$$\frac{\partial L}{\partial W} = \frac{\partial J}{\partial W} + \Lambda^T \frac{\partial R}{\partial W} \overset{!}{=} 0$$

Adjoint state equation

$$\frac{\partial L}{\partial \phi} = \frac{\partial J}{\partial \phi} + \Lambda^T \frac{\partial R}{\partial \phi} \overset{!}{=} 0$$

Design equation

Optimality System

- **Optimization Problem:**

$$\min_{\phi \in \Phi} J(W,\phi) \quad s.t. \quad R(W,\phi) = 0 \quad \leftrightarrow$$

Fixed point iteration:
$$W = G(W,\phi)$$

- **Lagrangian: instead** $L = J + \Lambda^T R$**, continuous or discrete** L**:**

$$L = J + \langle \Lambda, R \rangle_{H^*,H} \quad \leftrightarrow \quad L(W,\Lambda,\phi) = J(W,\phi) + \Lambda^T (G(W,\phi) - W)$$

- **Optimality condition (KKT system, 1. order necessary cond.):**

Black-box differentiation:

$$\frac{\partial L}{\partial \Lambda} = G(W,\phi) - W \overset{!}{=} 0$$
State equation

$$\frac{\partial L}{\partial W} = \frac{\partial J}{\partial W} + \Lambda^T \left(\frac{\partial G}{\partial W} - I \right) \overset{!}{=} 0 \Leftrightarrow N_W^T(W,\Lambda,\phi) = \Lambda$$
Adjoint state equation

$$\frac{\partial L}{\partial \phi} = \frac{\partial J}{\partial \phi} + \Lambda^T \frac{\partial G}{\partial \phi} \overset{!}{=} 0$$
Design equation

Optimality System

- **Optimization Problem:**

$$\min_{\phi\in\Phi} J(W,\phi) \quad s.t. \quad R(W,\phi)=0 \quad \Leftrightarrow$$

Fixed point iteration:
$$W = G(W,\phi)$$

- **Lagrangian: instead** $L = J + \Lambda^T R$**, continuous or discrete** L**:**

$$L = J + \langle \Lambda, R \rangle_{H^*,H} \quad \Leftrightarrow \quad L(W,\Lambda,\phi) = J(W,\phi) + \Lambda^T (G(W,\phi) - W)$$

Black-box differentiation:

$$G(W,\phi) - W \overset{!}{=} 0$$

$$\frac{\partial J}{\partial W} + \Lambda^T \left(\frac{\partial G}{\partial W} - I \right) \overset{!}{=} 0 \Leftrightarrow N_W^T(W,\Lambda,\phi) = \Lambda$$

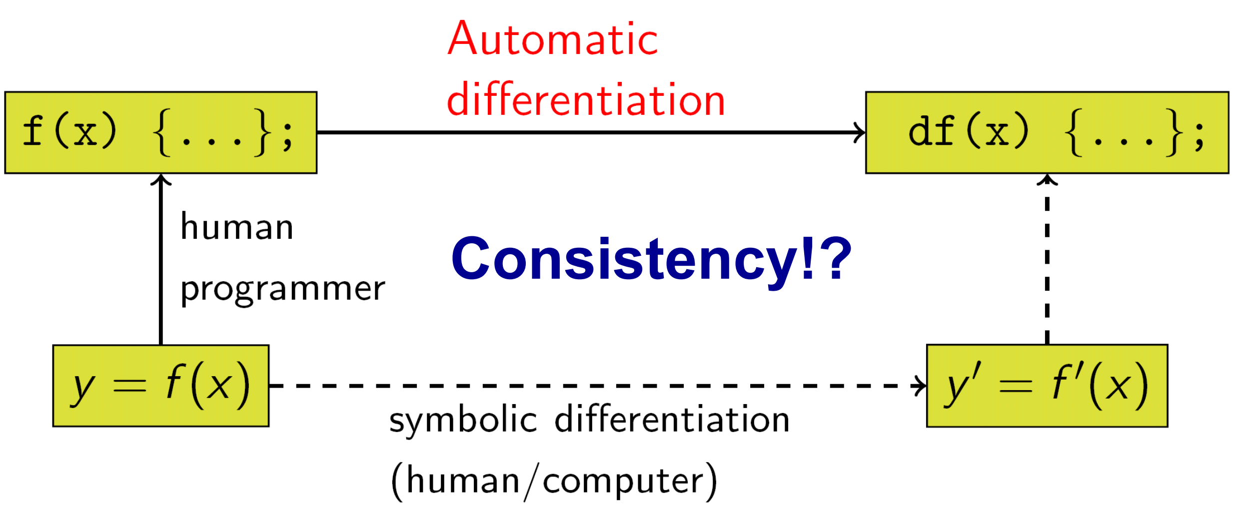
Primal contractivity: $\left\| G_W \right\| = \left\| G_W^T \right\| \le \rho < 1 \Rightarrow$ Adjoint contractivity:

$$\left\| \frac{\partial N_W^T}{\partial \Lambda} \right\| = \left\| G_W^T \right\| \le \rho < 1$$

Adjoint code inherits convergence properties of primal code

Discrete Adjoint Method

- ***First discretise then optimise***
  - ***First discretise the primal system***
  - ***Then obtain the adjoint system based on the discretised primal equations***
- ***Possibility proposed here: Automatic or Algorithmic Differentiation (AD)***
- ***Basic principle***
  - ***Computer code is concatenation of basic operations (+,-,*,etc.)***
  - ***Apply differentiation rules to this concatenation by using chain rule***



Semi-automatic Transition from Simulation to Optimization

Numerical Discretization

Meta-Programming Techniques

Fixed-Point Formulation

Algorithmic Differentiation

Semi-automatic generation

Discrete Adjoint Solver

Consistency

Exact derivatives based on the particular numerical methods.

Robustness

Inheritance of convergence properties.

Efficiency

Competitive overall performance.

Fixed-Point Formulation for Multi-Disciplinary Design

- $\beta \in \mathcal{D} \subset \mathbb{R}^p$: design vector
- $U \in \mathcal{U} \subset \mathbb{R}^n$: state vector
- $X \in \mathcal{X} \subset \mathbb{R}^m$: computational mesh
- $M(\beta) = X$: mesh deformation equation
- $J(U, X)$: objective function
- $R(U, X) = 0$: discretized state equation

R or rather G may not only contain the flow equation, but **any coupled model**, i.e.

- (U)RANS + turbulence (→ working)
- (U)RANS + structure (→ in development)
- (U)RANS + aeroacoustics (→ working)

and **arbitrary boundary conditions**.

$$\begin{aligned} \min_\beta \quad & J(U(\beta), X(\beta)) & \\ \text{s.t.} \quad & R(U(\beta), X(\beta)) &= 0 \\ & M(\beta) &= X \end{aligned}$$

Assuming $R(U, X) = 0$ is solved by a fixed-point iteration: $G(U^*, X) = U^* \Leftrightarrow R(U^*, X) = 0$

$$\begin{aligned} \min_\beta \quad & J(U(\beta), X(\beta)) & \\ \text{s.t.} \quad & G(U(\beta), X(\beta)) &= U \\ & M(\beta) &= X \end{aligned}$$

In case of Newton-type solver: $G(U, X) := U - P(U, X)R(U, X)$, where $P \approx (\partial R/\partial U)^{-1}$.

The Discrete Adjoint Solver

Using the method of Lagrangian multiplier we define the **Lagrangian function** as:

$$L(\beta, U, X, \bar{U}, \bar{X}) \quad = \quad J(U,X) + \bar{U}^T(G(U,X) - U) + \bar{X}^T(M(\beta) - X)$$

KKT conditions yield equations for adjoints $\bar{U}, \bar{X}$ and sensitivity vector $dL/d\beta$:

$$\begin{aligned}
\bar{U} &= \frac{\partial}{\partial U} J(U,X) + \frac{\partial}{\partial U} G^T(U,X)\bar{U} \\
&= \frac{\partial}{\partial U} N^T(U,\bar{U},X) \qquad \textbf{Adjoint equation} \\
\bar{X} &= \frac{\partial}{\partial X} J(U,X) + \frac{\partial}{\partial X} G^T(U,X)\bar{U} \\
&= \frac{\partial}{\partial X} N^T(U,\bar{U},X) \qquad \textbf{Mesh Adjoint equation} \\
\frac{dL}{d\beta} &= \frac{d}{d\beta} M^T(\beta)\bar{X} \qquad \textbf{Design equation}
\end{aligned}$$

Duality-Preserving and Simplified Iteration

By construction, the iteration

$$\bar{U}^{n+1} = \frac{\partial}{\partial U} J(U^*, X) + \frac{\partial}{\partial U} G^T(U^*, X)\bar{U}^n$$

to solve the adjoint equation is **Duality-Preserving**[1,2](i.e. it satisfies the duality condition and its convergence rate is given by the dominant eigenvalue of $\frac{\partial}{\partial U} G^T(U, X)$).

Note that in case of a **Newton-type solver** for the derivative of G with respect to U holds

$$\frac{\partial G}{\partial U} = I - P^{-1}\frac{\partial R}{\partial U} - \left[\frac{\partial P}{\partial U}\right]^{-1} R,$$

We may drop the last term and use the **Simplified Iteration**

$$\bar{U}^{n+1} \quad = \quad \frac{\partial J}{\partial U^*} + \left(I - \left[\frac{\partial R}{\partial U^*}\right]^T P^{-T}\right) \bar{U}^n$$

[1] E.J. Nielsen, J. Lu, M.A. Park, D.L. Darmofal. *An exact dual adjoint solution method for turbulent flows on unstructured grids.* AIAA Paper 2003-0272, 2003

[2] D.J. Mavriplis. *Multigrid solution of the discrete adjoint for optimization problems on unstructured meshes.* AIAA Journal 44.1, 2006

Expression templates

Consider the statement

$$w = ((a + b) * (c - d))^2.$$

Expression templates

Consider the statement

$$w = ((a + b) * (c - d))^2.$$

The right-hand side expression is represented by the (static) object

POW<MULT<ADD<ActiveReal, ActiveReal>, SUB<ActiveReal, ActiveReal>>>

Expression templates

Consider the statement

$$w = ((a + b) * (c - d))^2.$$

The right-hand side expression is represented by the (static) object

POW<MULT<ADD<ActiveReal, ActiveReal>, SUB<ActiveReal, ActiveReal>>>

This "generates" the code:

```
add = a + b; sub = c - b; mul = add * sub;
w = pow(mul, 2.0);

w_b = 1.0;
mul_b += 2 * mul * w_b;
sub_b += add * mul_b;
add_b += sub * mul_b;
c_b += sub_b;
d_b += -sub_b;
a_b += add_b;
b_b += add_b;
tape.pushJacobi(a_b, a.index);
tape.pushJacobi(b_b, b.index);
tape.pushJacobi(c_b, c.index);
tape.pushJacobi(d_b, d.index);
tape.pushStatement(w.index, 4);
```

Expression templates

Consider the statement

$$w = ((a + b) * (c - d))^2.$$

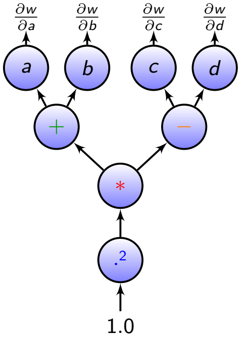
The right-hand side expression is represented by the (static) object

POW<MULT<ADD<ActiveReal, ActiveReal>, SUB<ActiveReal, ActiveReal>>>

This "generates" the code:

```
add = a + b; sub = c − b; mul = add * sub;
w = pow(mul, 2.0);

w_b = 1.0;
mul_b += 2 * mul * w_b;
sub_b += add * mul_b;
add_b += sub * mul_b;
c_b += sub_b;
d_b += −sub_b;
a_b += add_b;
b_b += add_b;
tape.pushJacobi(a_b, a.index);
tape.pushJacobi(b_b, b.index);
tape.pushJacobi(c_b, c.index);
tape.pushJacobi(d_b, d.index);
tape.pushStatement(w.index, 4);
```



Expression templates

Consider the statement

$$w = ((a + b) * (c - d))^2.$$

The right-hand side expression is represented by the (static) object

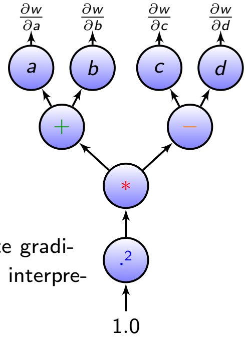
POW<MULT<ADD<ActiveReal, ActiveReal>, SUB<ActiveReal, ActiveReal>>>

This “generates” the code:

```
add = a + b; sub = c − b; mul = add ∗ sub;
w = pow(mul, 2.0);

w_b = 1.0;
mul_b += 2 ∗ mul ∗ w_b;
sub_b += add ∗ mul_b;
add_b += sub ∗ mul_b;
c_b += sub_b;
d_b += −sub_b;
a_b += add_b;
b_b += add_b;
tape.pushJacobi(a_b, a.index);
tape.pushJacobi(b_b, b.index);
tape.pushJacobi(c_b, c.index);
tape.pushJacobi(d_b, d.index);
tape.pushStatement(w.index, 4);
```

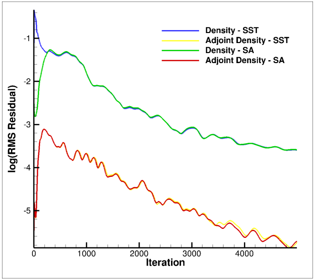
Data used to accumulate gradients in a second reverse interpretation step

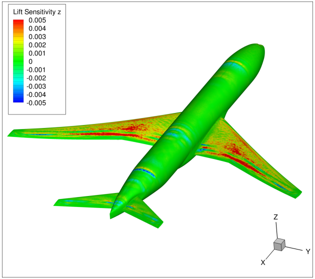


Further information

- Gradient evaluation using Expression templates implemented in the CoDiPack C++ software library
  - developed with focus on industrial and HPC applications
  - different taping strategies
  - dynamic source-code analysis features
  - vector-mode and higher order derivatives
  - released under GPL3, available on Github
- Requires (almost) **no** modifications in the flow (or coupled) solver
- No special coding rules for developers
- **Easily extensible** to incorporate new models and objective functions
- **Parallelized** using AdjointMPI (will be replaced by a new library soon)

NASA Common Research Model





Goal: Redesign of the CRM wing and tail at transonic flight conditions

- Flow solution and sensitivities computed using the direct and discrete adjoint solver in SU2
- 10 Million elements for discretization (coarse mesh)
- 112 Cores on cluster Elwetritsch@TUKL for direct and adjoint
- $\frac{\text{Wall-clock time adjoint}}{\text{Wall-clock time direct}} = \frac{4.3\text{ h}}{3.7\text{ h}} = 1.2$
- $\frac{\text{Total memory adjoint}}{\text{Total memory direct}} = \frac{710\text{ GB}}{98\text{ GB}} = 7.2$

Joint work with
Prof. J. Alonso, T. Economon, Stanford University
F. Palacios, J. Vassberg, The Boeing Company

A Coupled CFD-CAA Framework for Noise Prediction

A 2-D frequency-domain permeable surface Ffowcs Williams-Hawkings (FW-H) acoustic solver [Lockard, 2000] is coupled with the URANS solver in SU2 for efficient far-field noise computations at arbitrary observer locations.

Permeable surface FW-H:

$$\left(\frac{\partial^2}{\partial t^2} + U_i U_j \frac{\partial^2}{\partial y_i \partial y_j} + 2U_i \frac{\partial^2}{\partial y_i \partial t} - c_\infty^2 \frac{\partial^2}{\partial y_i \partial y_i}\right) \left[\rho' H(f)\right] = \frac{\partial}{\partial t}\left[Q\delta(f)\right] - \frac{\partial}{\partial y_i}\left[F_i \delta(f)\right] + \frac{\partial^2}{\partial y_i \partial y_j}\left[T_{ij} H(f)\right] \tag{1}$$

where

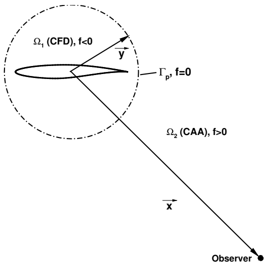
$$Q(\vec{\mathbf{y}}, t) = \left[\rho\left(u_i' + U_i\right) - \rho_\infty U_i\right] \hat{n}_i$$
$$F_i(\vec{\mathbf{y}}, t) = \left[\rho\left(u_i' - U_i\right)\left(u_j' + U_j\right) + \rho_\infty U_i U_j + p\delta_{ij} - \tau_{ij}\right] \hat{n}_i$$
$$T_{ij}(\vec{\mathbf{y}}, t) = \rho u_i' u_j' + \left[p - c_\infty^2(\rho - \rho_\infty)\right] \delta_{ij} - \tau_{ij}.$$

Q, F_i and T_{ij} are monopole, dipole and quadrupole source terms respectively. Transforming into frequency domain and convoluting with a free space Green's function...

A Coupled CFD-CAA Framework for Noise Prediction

Convoluting with free space Green's function, we obtain the boundary integral form of the permeable surface FW-H in frequency domain:

$$\tilde{p}'_{obs}(\vec{\mathbf{x}},\omega) = -\oint_{f=0} i\omega\tilde{Q}(\vec{\mathbf{y}},\omega)G(\vec{\mathbf{x}},\vec{\mathbf{y}},\omega)\,dl - \oint_{f=0}\tilde{F}_i(\vec{\mathbf{y}},\omega)\frac{\partial G(\vec{\mathbf{x}},\vec{\mathbf{y}},\omega)}{\partial y_i}\,dl - \int_{f>0}\tilde{T}_{ij}(,\vec{\mathbf{y}},\omega)\frac{\partial^2 G(\vec{\mathbf{x}},\vec{\mathbf{y}},\omega)}{\partial y_i \partial y_j}\,dl \tag{2}$$



where

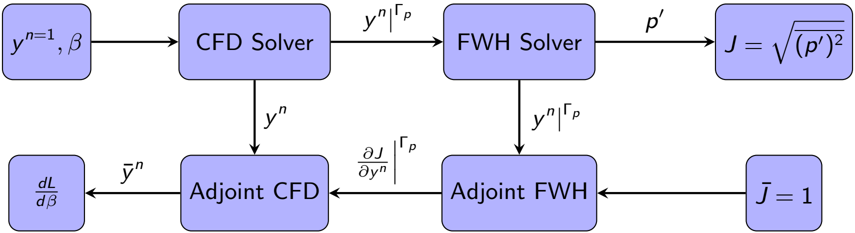
$\tilde{Q}$, $\tilde{F}_i$ and $\tilde{T}_{ij}$ are fourier-transformed source terms

G is an analytically known Green's function

- Flow field in Ω_1 resolved by CFD
- p, ρ, u'_i on Γ_p extracted from CFD data
- $\tilde{p}'_{obs}$ computed from line integral (2)
- $\tilde{p}'_{obs}(\vec{\mathbf{x}},\omega) \rightarrow p'_{obs}(\vec{\mathbf{x}},t)$ using iFFT

AD-based discrete adjoint: well suited to such multi-module computational design chain – differentiates through the coupled program *algorithmically* without cumbersome treatment at module interfaces.

Coupled CFD-FWH Noise Prediction and Optimization Framework



- CFD Solver: $y^n = G^n(y^n, y^{n-1}, y^{n-2})$, [Eqn 3]
- FWH Solver: $\tilde{p}'_{obs} = -\oint_{f=0} i\omega \tilde{Q} G dl - \oint_{f=0} \tilde{F}_i \frac{\partial G}{\partial y_i} dl$, [Eqn 2]
- Adjoint CFD: $\bar{y}^n = \bar{G}^n(\bar{y}^n, \bar{y}^{n-1}, \bar{y}^{n-2})$, [Eqn 4]
- $y^n|^{\Gamma_p}$: Flow variables at time step n on the FWH surface Γ_p
- $\frac{\partial J}{\partial y^n}\Big|^{\Gamma_p}$: sensitivity of the noise objective with respect to flow variables evaluated on the FWH surface Γ_p

Lift-Constrained Drag and Noise Minimization

Flow Conditions

- Baseline airfoil: NACA 64A010
- $M_\infty = 0.796$
- Pitching frequency $\omega_r = 0.202$, amplitude $A = 1^\circ$ and mean angle of attack $\alpha_{mean} = 0^\circ$, around 1/4 chord point

Numerical Settings

- Unstructured mesh with 16,937 triangular elements
- JST scheme + dual-time stepping
- Design variables: 50 (25+25) Hicks-Henne functions
- 25 Δt/period for 10 periods

Pitching Airfoil in Transonic Inviscid Flow

Case I – Lift-Constrained Drag Minimization

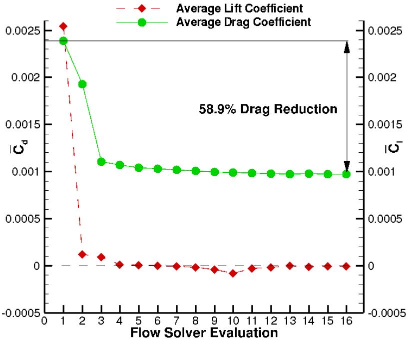
$$\begin{aligned} \min_{\beta} \quad f^D &= \frac{1}{N-N^*} \sum_{n=N^*}^{N} \hat{f}^n(y^n,\beta), \qquad \hat{f}^n = C_d^n \\ \text{subject to} \quad y^n &= G^n(y^n, y^{n-1}, y^{n-2}, \beta), \qquad n = 1,\ldots,N \\ \bar{C}_l &\geq 0 \\ Area &= Area_{baseline} \end{aligned}$$

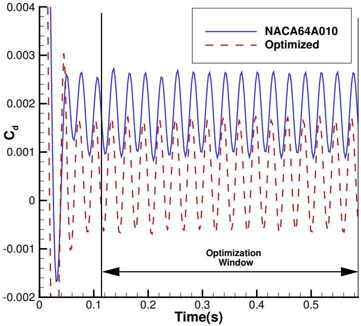
Case II – Lift-Constrained Noise Minimization

$$\begin{aligned} \min_{\beta} \quad f^N &= \frac{1}{N-N^*} \sum_{n=N^*+1}^{N} \hat{f}^n(y^n,\beta), \qquad \hat{f}^n = (p_{obs}^n - p_\infty)^2 \\ \text{subject to} \quad y^n &= G^n(y^n, y^{n-1}, y^{n-2}, \beta), \qquad n = 1,\ldots,N \\ \bar{C}_l &\geq 0 \\ Area &= Area_{baseline} \end{aligned}$$

(Similar study by Rumpfkeil & Zingg in 2010, without lift constraint)

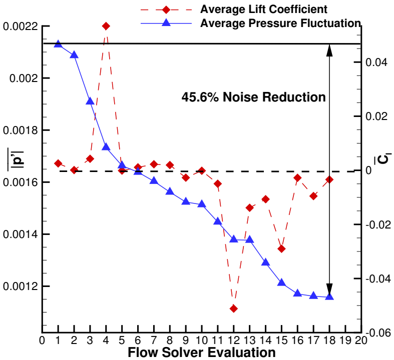
Lift-Constrained Drag Minimization

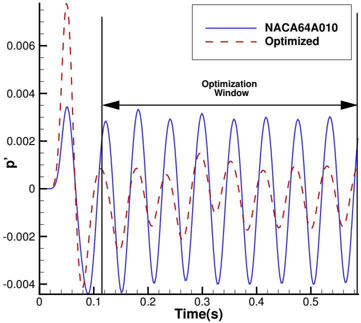




- Optimization performed over 16 periods of oscillation
- Time-averaged drag reduced by $\sim 59\%$ after 16 CFD evaluations
- Time-averaged lift maintained ~ 0

Lift-Constrained Noise Minimization

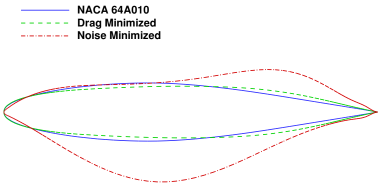




- A single observer point located $10c$ below the trailing edge
- Optimization performed over 8 periods of oscillation
- Time-averaged pressure fluctuation reduced by $\sim 46\%$

Comparison of Optimal Designs

	NACA64A010	Drag Minimized	Noise Minimized
f^D	2.39×10^{-3}	9.75×10^{-4} $(-59\%\ f_0^D)$	1.57×10^{-1}
f^N	2.13×10^{-3}	2.03×10^{-3}	1.16×10^{-3} $(-46\%\ f_0^N)$



- Drag and noise may be competing objectives
- Two optimizations, conducted in parallel, do not lead to the same shape
- Aeroacoustic consideration cannot be an 'after-thought' and must be included in the initial design process

Even though designs based on steady aerodynamics is still the industry standard today, efficient computational tools must be developed and refined to address inherently unsteady design problems such as noise reduction.

Overview of Adjoint-based Multi-Stage Optimization

- Frey, C., Kersken, H.-K., and Nrnberger, D., *The Discrete Adjoint of a Turbomachinery RANS Solver*", **2009**
- Wang, D., and He, L., *Adjoint Aerodynamic Design Optimization for Blades in Multistage Turbomachines: Part I/II*, **2010**
- Walther, B. and Nadarajah, S., "*Adjoint-Based Constrained Aerodynamic Shape Optimization for Multistage Turbomachines*", **2015**
- Yu, J., Ji, L., Li, W., and Yi, W., "*Adjoint Optimization of Multistage Axial Compressor Blades with Static Pressure Constraint at Blade Row Interface*", **2015**

Why is the application of the adjoint approach to multi-stage turbomachines lagging far behind ?

Apparent problem is the definition of appropriate adjoint boundary conditions for **periodic boundaries**, **non-reflecting boundaries**, **mixing-plane interfaces** etc ...

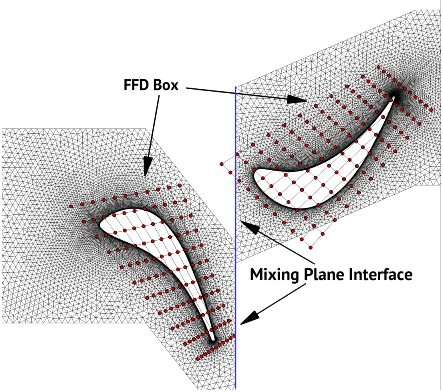
Inherent problem of continuous and (hand-) discrete adjoint methods.

Operator Overloading allows **rapid (simultaneous)** development of an adjoint solver in that case !

Aachen Turbine[1]

Minimize Entropy Production

$$\min_{\alpha} J(\alpha) := \sum_{i=1}^{2} \vartheta_i = \sum_{i=1}^{2} \frac{s_{i,in} - s_{i,out}}{s_{i,n}}$$

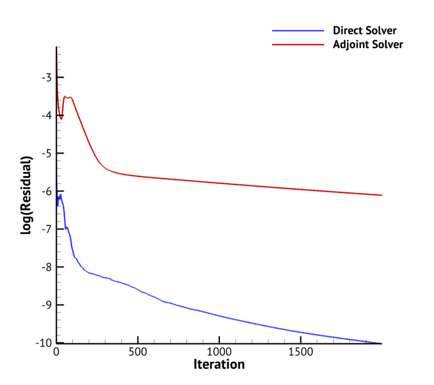


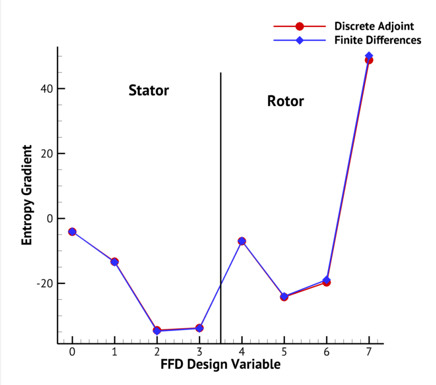
Mixing-Plane Performance

Outlet Density 1.54127	Inlet Density 1.54414
Outlet Pressure 124406	Inlet Pressure 124538
Outlet Normal Velocity 100.369	Inlet Normal Velocity 100.051
Outlet Tang. Velocity -197.801	Inlet Tang. Velocity -47.7791

[1] Walraevens, R., and Gallus, H. *Testcase 6 1-1/2 Stage Axial Flow Turbine.* ERCOFTAC SIG on 3D Turbomachinery Flow Prediction, 2000

Convergence and Validation

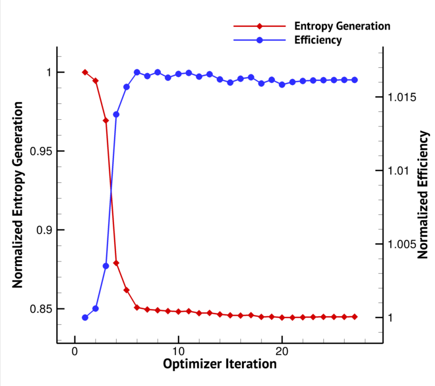


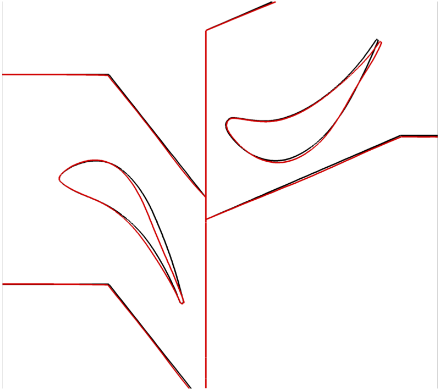


Note on performance:

$$\frac{\text{Time for adjoint solution}}{\text{Time for flow solution}} = 1.0, \frac{\text{Memory Adjoint Solver}}{\text{Memory Flow Solver}} = 4.5$$

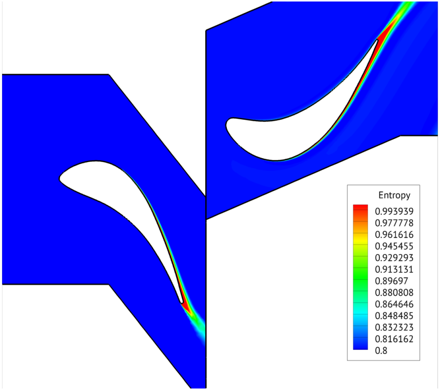
Optimization History



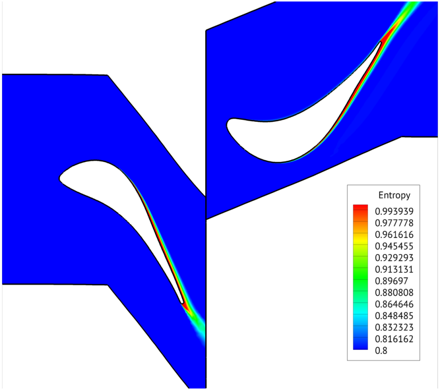


Entropy Field

Baseline



Optimized



Thanks for your attention!