Scientific Computing

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

CHT problems and computing coupled discrete adjoints using AD

Ole Burghardt, Tim Albring and Nicolas R. Gauger

Chair of Scientific Computing
TU Kaiserslautern, Germany

September 17, 2018

1 Introductory example

2 Computing accurate (discrete) adjoints in SU2

3 Coupled problems and validations

Conjugate heat transfer (CHT) problems

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

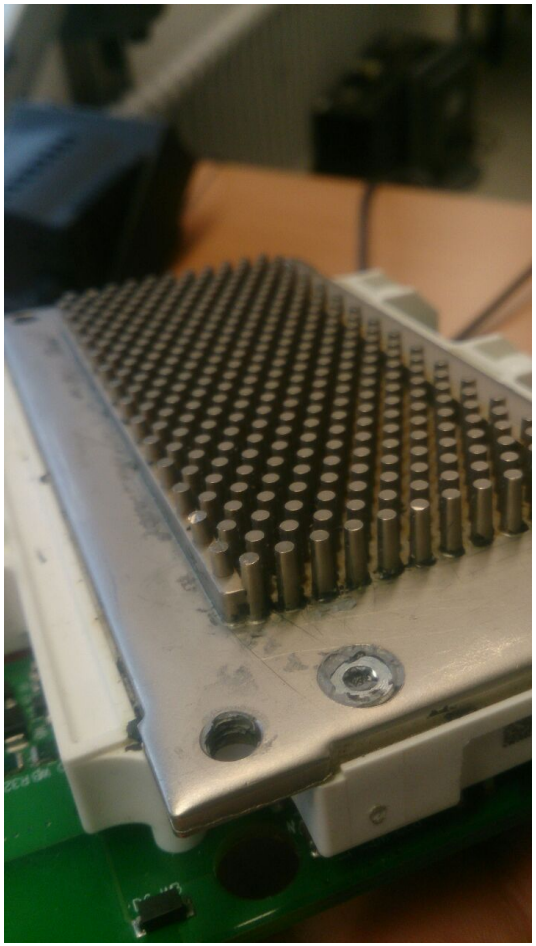
Problem characteristics:

- multiple physical zones that exchange heat at some interfaces
- different governing equation sytems, conductivities, timesteps, ...

Why to start developments in SU2?

- approach from a Bosch research department working on pin-fin cooler optimizations
- joint project with them since 2017

Can SU2 give accurate sensitivities?



Forward (primal) implementation

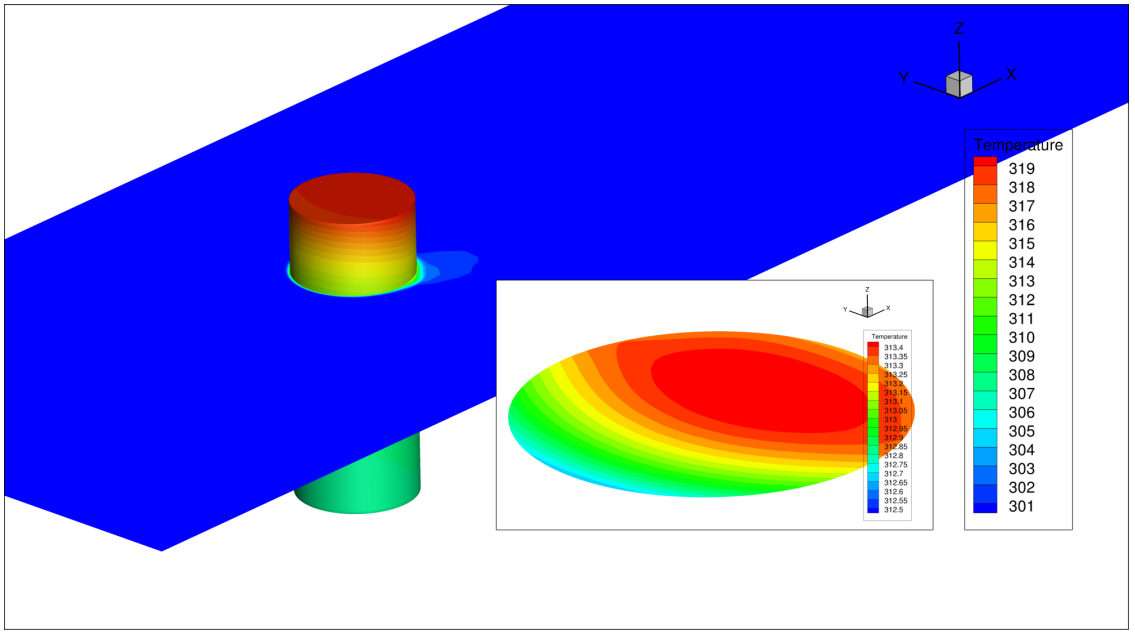
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Required ...

- A heat solver (cf. `CHeatSolverFVM`) that can be run in solid zones and coupled to `CIncNSSolver`
- Transfer routines to communicate energies (cf. `CTransfer_ConjugateHeatVars`-class)
- A new driver to iterate the solvers with this kind of coupling (ended up in a much more general approach to handle arbitrary ones, cf. `CMultiphysicsZonalDriver`)

3D test case

Heated aluminium cylinder in (coolant) water flow (at $0,25\frac{m}{s}$, 300K at inlet, 4W heat load at pin's top).



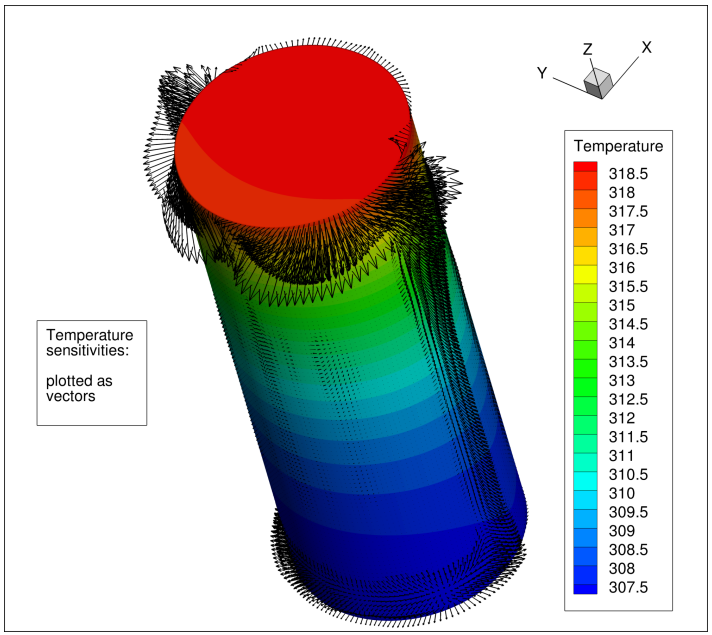
Pin's height/diameter: 5mm/2mm. **Reaches 319K** in average at its top in good agreement to FLUENT.

Optimization with shape gradients

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Define

- X be the vector of mesh node *coordinates* determining the pin's surface
- an objective function $J(X)$, e.g. the average temperature at a pin's tip



$\nabla J(X^*)$ suggests the geometry update for minimization of J. It should be passed

- to an appropriate shape update function (FFD, filtering, ...)
- and an optimizer.

How to obtain such sensitivities?

$J(X)$ depends – more precisely – on a state vector U holding pressure, momentum and temperature solutions at each node,

$$J(X) = \tilde{J}(X, U(X)).$$

Denote by $\mathcal{G}$ a CFD solver iterating the vectors U_k, that is

$$U_{k+1} = \mathcal{G}_{(X)}(U_k).$$

The solution $U(X)$ is given by the first iterate solution fulfilling

$$\|\mathcal{G}_{(X)}(U) - U\| < \varepsilon.$$

The computation cost of $\nabla J(X)$ crucially depends on the complexity of $\mathcal{G}$.

Rewriting $J(X)$ as a Lagrangian

Regard J and $\mathcal{G}$ as functions of *two* vectors X, U and set up the **Lagrangian**

$$L(X, U) = \tilde{J}(X, U) + (\tilde{\mathcal{G}}(X, U) - U)^T \cdot \lambda.$$

Along actual flow solutions, L equals $\tilde{J}$, independent of the choice of λ.

By λ we denote the factor („Lagrange multiplier“) such that

$$\nabla_U \tilde{J}(X, U) = (D_U \tilde{\mathcal{G}}^T(X, U) - \mathrm{Id}) \cdot \lambda.$$

This condition ensures that at flow solutions we have

$$\nabla J(X) = \nabla_X L(X, U)$$

which is cheap to compute as it **does not involve the flow solver anymore**.

Method in SU2: fixed point iteration for λ

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

To obtain λ, we carry out the fixed point iteration

$$\lambda \stackrel{!}{=} \nabla_U \tilde{J}(X, U) + D_U \tilde{\mathcal{G}}^T(X, U) \cdot \lambda$$

Note that so far we didn't make any assumptions on $\mathcal{G}$ except for being able to carry out the derivative

$$D_U \tilde{\mathcal{G}}^T(X, U) \cdot \lambda$$

subject to an arbitrary vector λ to compute $\nabla J(X)$.

Use of AD (in reverse mode)

For an arbitrary implementation of a function $f : \mathbb{R}^n \to \mathbb{R}^n$ and an arbitrary vector $y \in \mathbb{R}^n$, automatic differentiation in reverse mode computes

$$Df^T(\tilde{x}) \cdot \lambda$$

by the following steps:

- registering the input variables x of the computer program f
- recording a tape while running f at a given point $\tilde{x}$, that is storing $Df(\tilde{x})$
- setting the values λ
- evaluating $Df^T(\tilde{x}) \cdot \lambda$

How to use this generality for multiphysics?

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Let our iterator $\mathcal{G}$ actually consist of two **coupled iterators**, $\begin{pmatrix}\mathcal{G}^A\\ \mathcal{G}^B\end{pmatrix}$, with two sets of meshes X, Y and variables U^A, U^B being iterated like

$$U_k^A \mapsto \mathcal{G}^A(X, U_k^B, U_k^A) = U_{k+1}^A$$
$$U_k^B \mapsto \mathcal{G}^B(Y, U_k^A, U_k^B) = U_{k+1}^B.$$

We then compute $\nabla J(X, Y)$ by simply computing both λ^A and λ^B in the corresponding Lagrangian

$$\tilde{J}(X, Y, U^A, U^B) + \begin{pmatrix}\mathcal{G}^A(X, U^B, U^A) - U^A\\ \mathcal{G}^B(Y, U^A, U^B) - U^B\end{pmatrix} \cdot \begin{pmatrix}\lambda^A\\ \lambda^B\end{pmatrix}.$$

Taking care of the cross terms

The fixed point interation for obtaining $\begin{pmatrix} \lambda^A \\ \lambda^B \end{pmatrix}$ now contains cross derivatives:

$$\begin{pmatrix} \lambda^A_{k+1} \\ \lambda^B_{k+1} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial U^A}\mathcal{G}^A & \frac{\partial}{\partial U^A}\mathcal{G}^B \\ \frac{\partial}{\partial U^B}\mathcal{G}^A & \frac{\partial}{\partial U^B}\mathcal{G}^B \end{pmatrix} \cdot \begin{pmatrix} \lambda^A_k \\ \lambda^B_k \end{pmatrix}$$

To keep the multizone discrete adjoint driver as **modular** as possible, the implementation allows for:

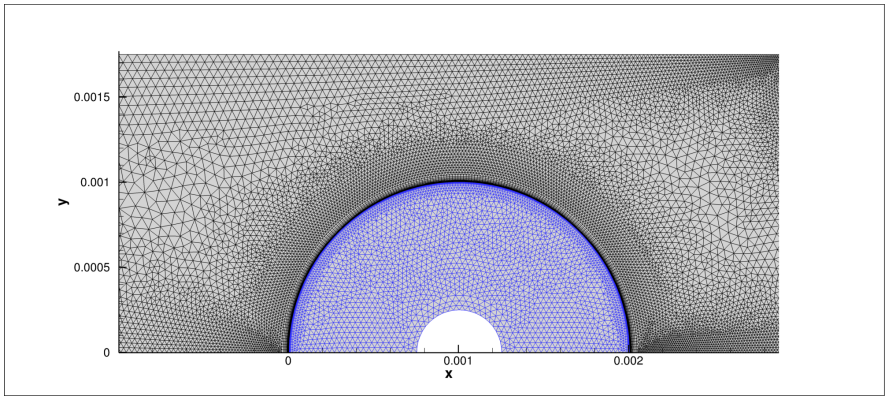
- initialising only parts of the right hand side adjoint vector
- evaluating with respect to arbitrary variable vectors
- restricting the evaluation to parts of the computational graph.

This provides a fair functionality for further **stable and efficient** developments.

Cylinder test case

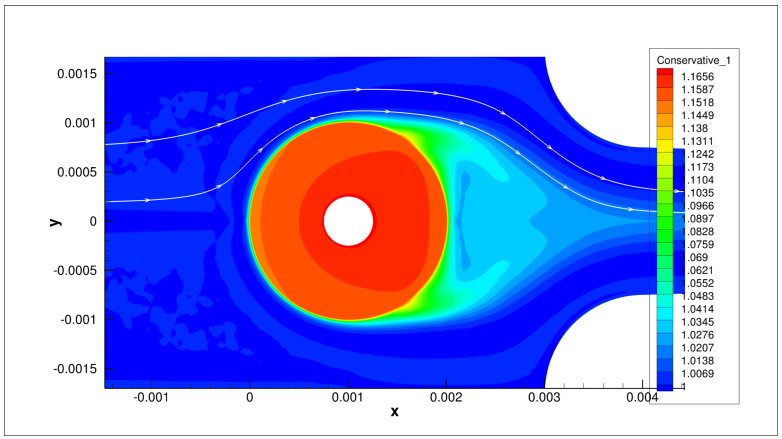
Let $\mathcal{G}^A$ be a **RANS solver** (with coupled heat equation) and $\mathcal{G}^B$ a **heat solver**, both coupled by transferring temperature and heat flux data.

The test geometry (denoted by X^*) is a cylinder, heated from the inside and surrounded by a fluid flow:



- Water, $300K$ / $0,25\frac{m}{s}$ at inlet
- Fixed conductivity ($\mathrm{Pr} = 7$)
- Aluminium
- Heat load at inner wall: $4\frac{kW}{m}$

Primal solve and objective function



- Obtained flow solution U^*
- Using averaged temperature at interface as objective function
- $\tilde{J}(X^*, U^*) = 415K$

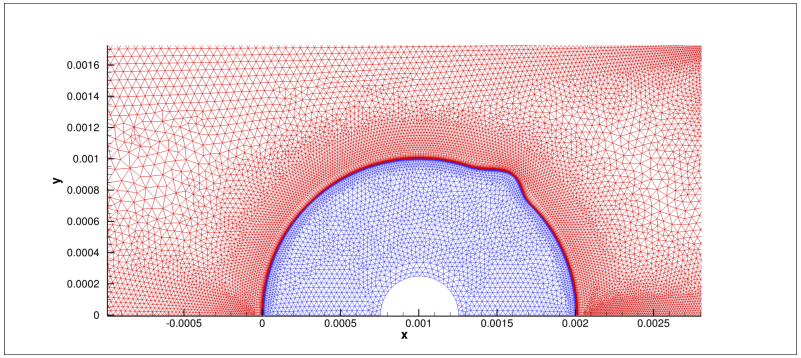
We easily compute the value

$$\nabla J(X) = \nabla_X \tilde{J}(X^*, U^*) + D_X \mathcal{G}^T(X^*, U^*) \cdot \lambda.$$

with the help of λ. **To validate ...**

Geometry change to validate $\nabla J(X)$

- we suggest a new geometry $Y_h = X + h\delta$
- $J(Y) \approx J(X) + h\nabla J(X) \cdot \delta$
- check $\lim_{h \to 0} \frac{J(Y_h) - J(X)}{h} \stackrel{?}{=} \nabla J(X) \cdot \delta$



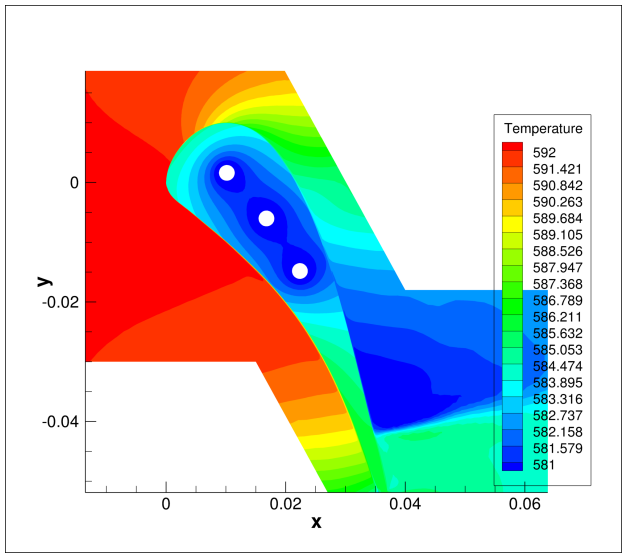
Changing the value of h, we obtain the following data:

h (in mm)	$J(Y)$ (in K)	$\frac{J(Y)-J(X)}{h}$ (in $\frac{K}{m}$)	rel. error to $\nabla J(X) \cdot \delta$
$1.0e^{-1}$	$417,873795$	$-65,89$	$8,68\%$
$5.0e^{-2}$	$417,877298$	$-61,74$	$1,83\%$
$1.0e^{-3}$	$417,879767$	$-60,7$	$1,15\%$
$5.0e^{-3}$	$417,880081$	$-60,6$	$0,05\%$

Feature work

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

- Uniting the CHT and turbomachinery functionalities for cooled turbine blade optimizations



- Speed up existing coupled simulations
- Trying out vertex-morphing optimizers, especially for internal flow applications