Topological Design Based on Highly Efficient Adjoints Generated by Automatic Differentiation

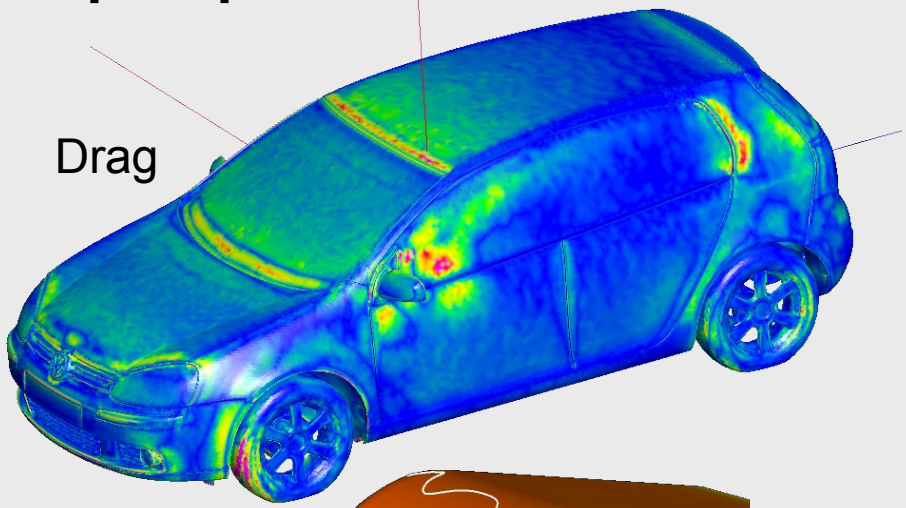
Thomas Kaminski[1], Ralf Giering[1], and Carsten Othmer[2]

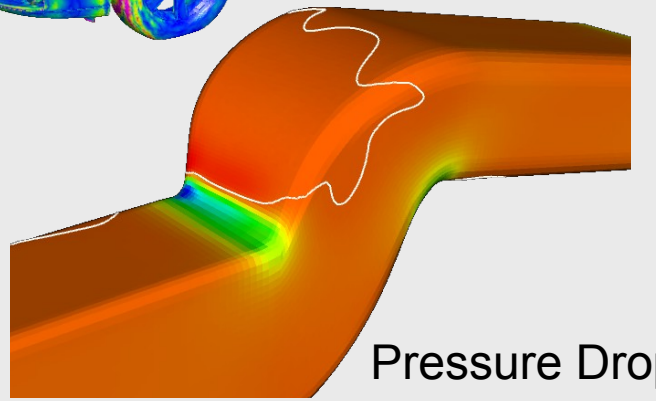
[1]**FastOpt** (http://FastOpt.com)
[2]**Volkswagen AG**

FastOpt

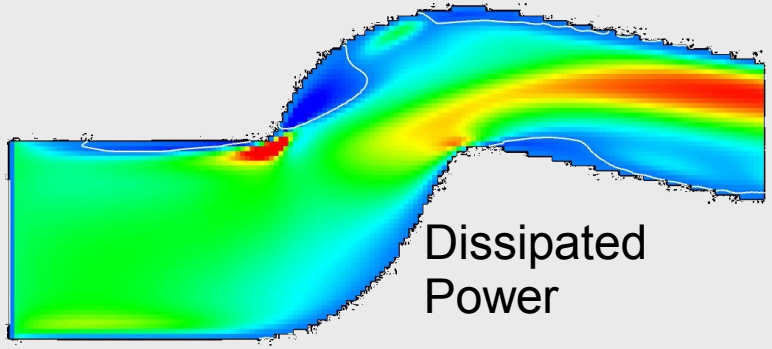
CFD optimisation with sensitivity maps: Examples

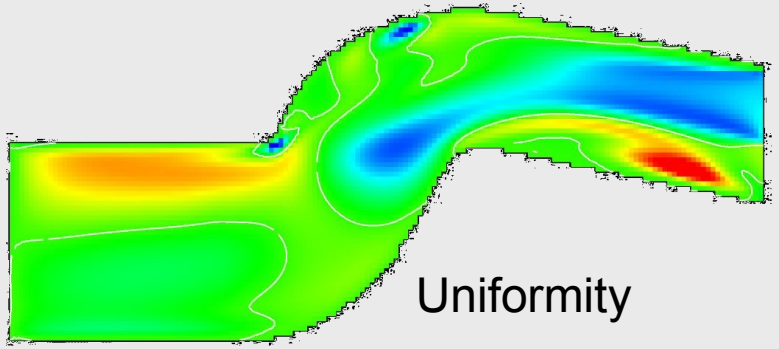
Shape Optimisation





Topology Optimisation

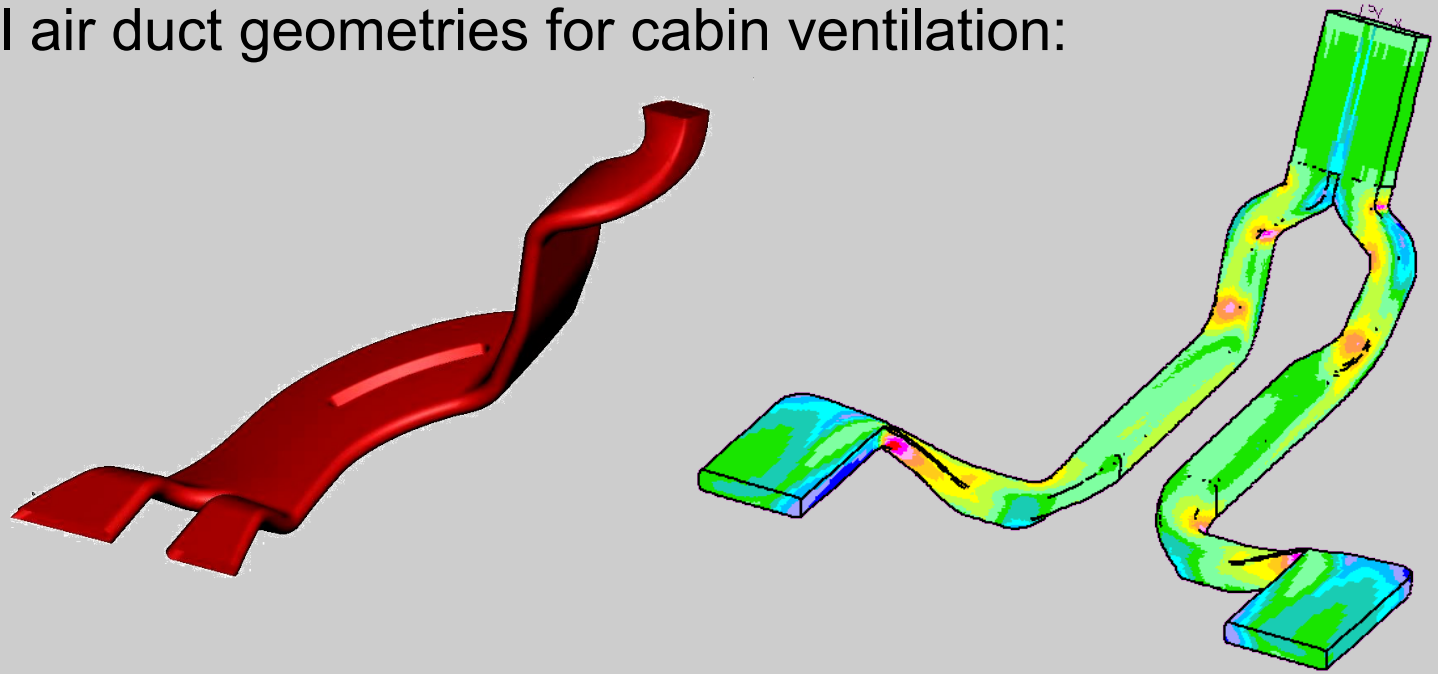




FastOpt

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

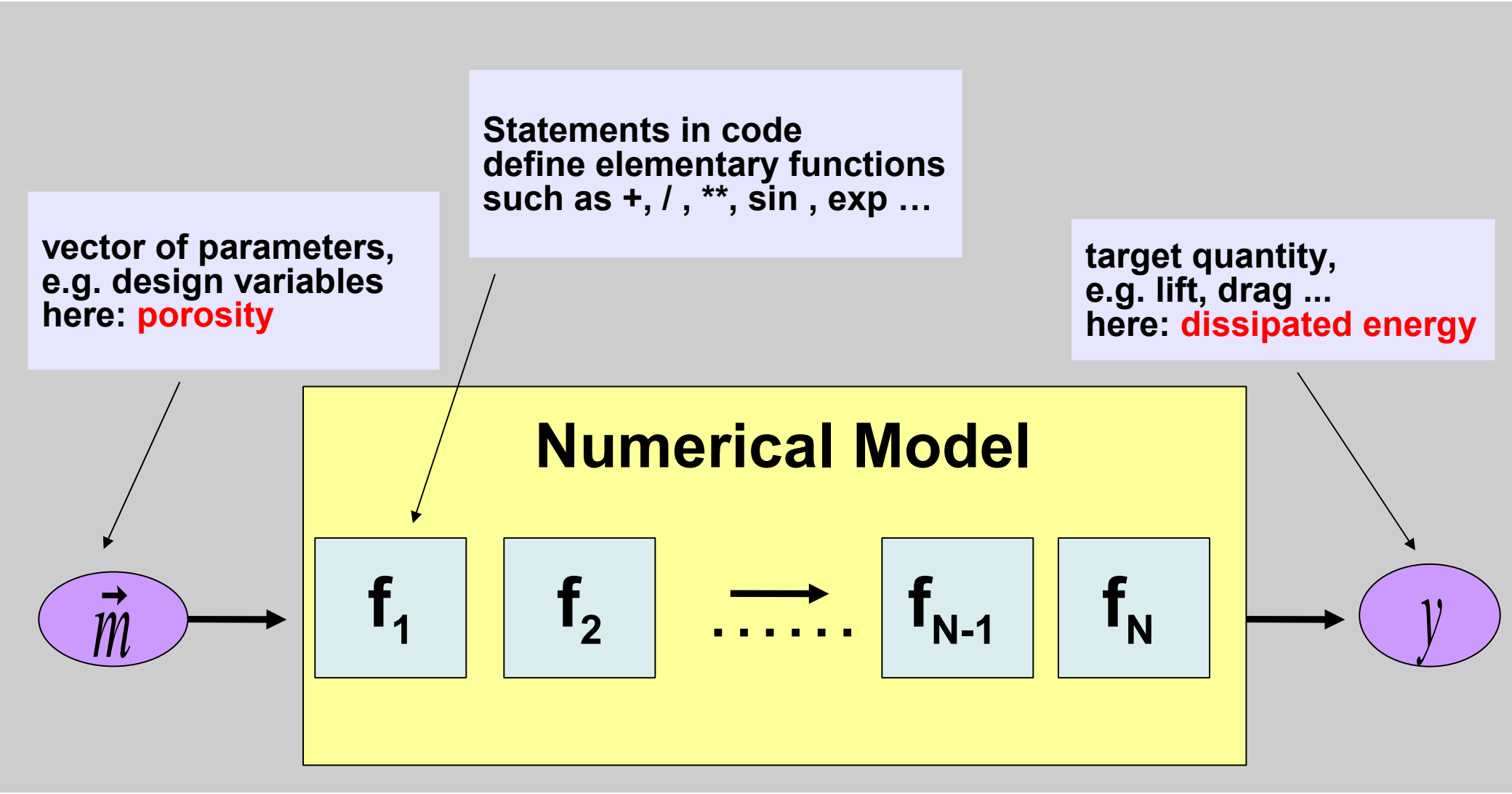
Topology optimization with adjoint methods

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 α_{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

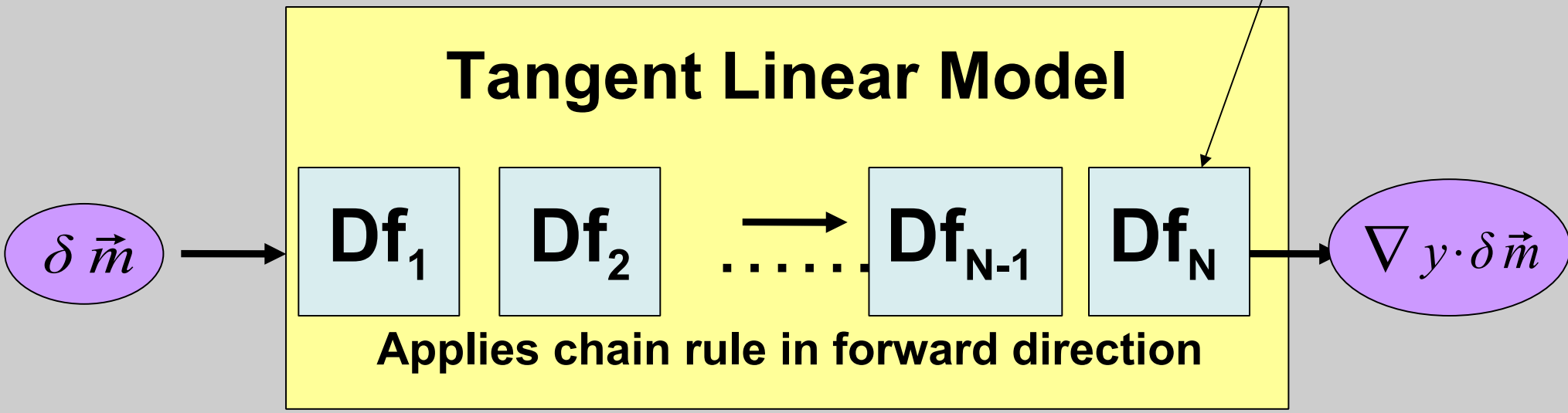
Sensitivities via AD



Sensitivities via AD: Tangent

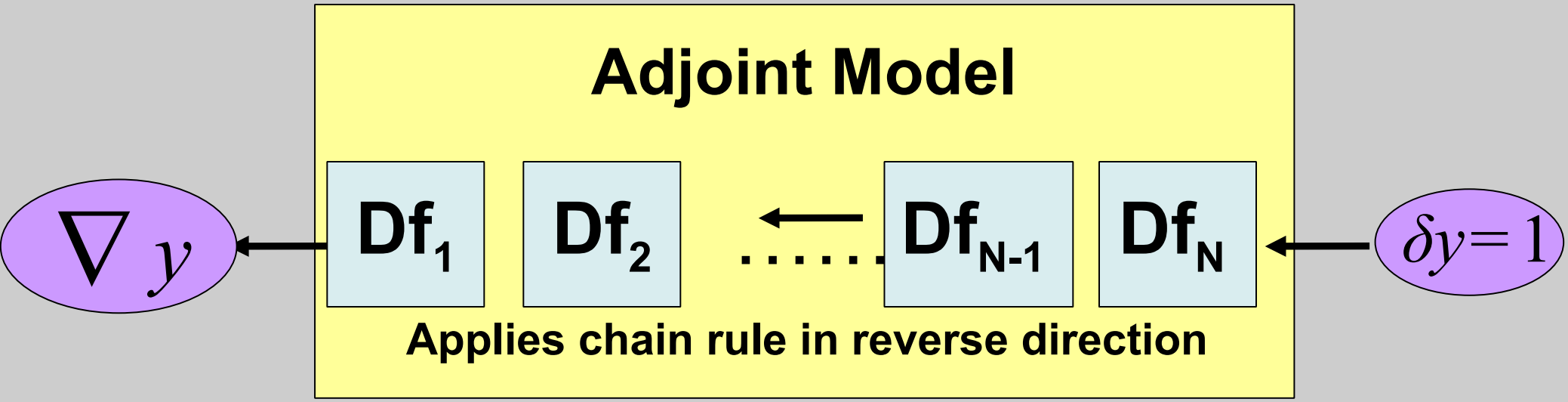
Cost of gradient evaluation proportional to # of parameters: One run of TLM per gradient component

Derivatives of elementary functions are simple, they define local Jacobians



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



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:

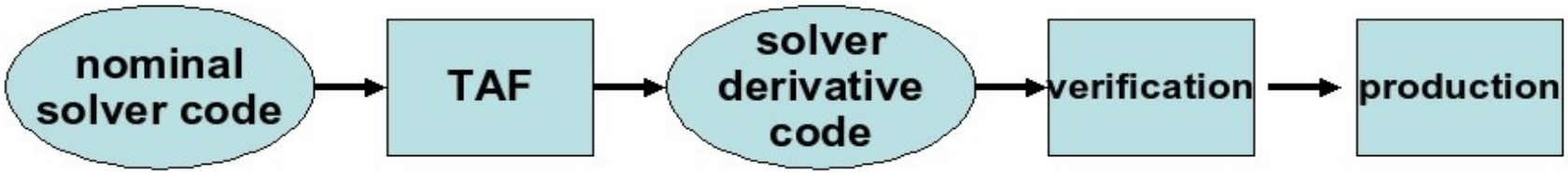
- Tangent linear code
- Adjoint code for steady problems (Giering et al., FGCS, 2005)
  linearises around converged primal flow
- Adjoint code for general problems
  suitable for time dependent or not fully converged problems stores sequence (history) of primal flows

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

	# of lines w/o comments	CPU (solve+grad)	rel acc. vs. FD
Primal	2700	1.0	
TLM	3300	1.3	1E-8
ADM steady	3700	1.8	1E-5
ADM general	3700	1.8	1E-8

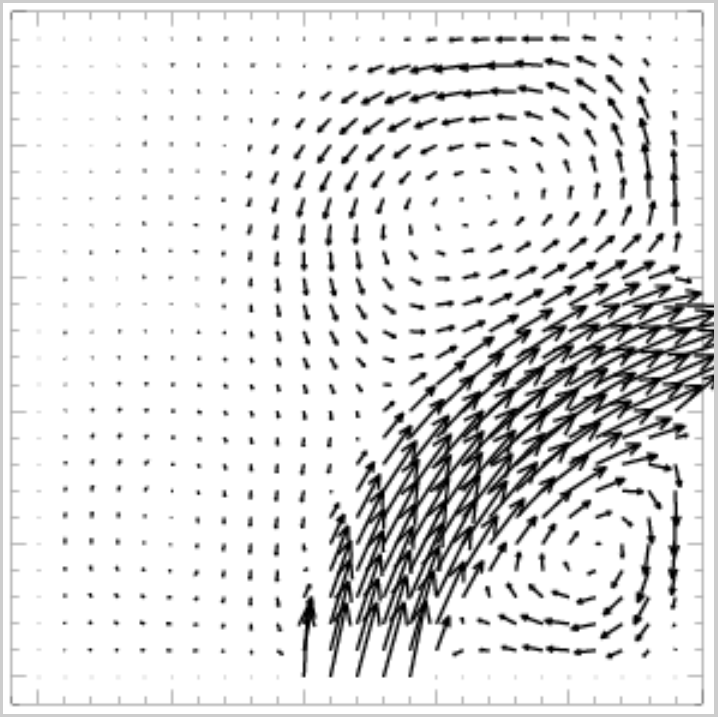
Automated Procedure

- 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

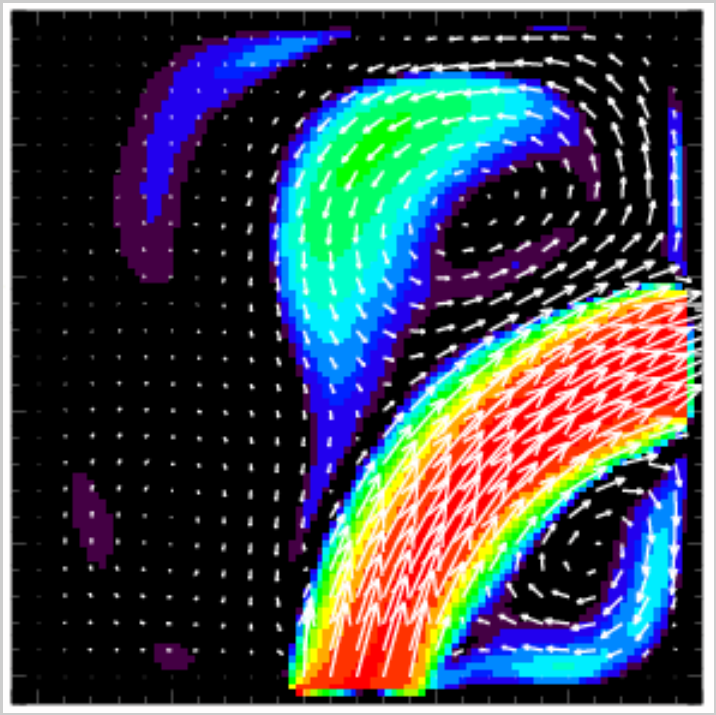
Velocity (Re=2500)



E=1.0

Adjoint approach: 2D laminar test case

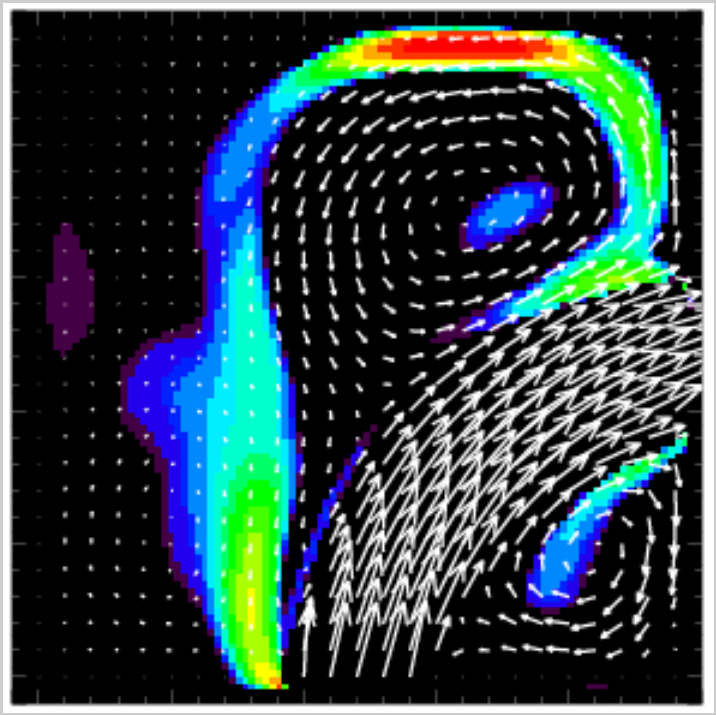
Sensitivities: “good“ cells



E=1.0

Adjoint approach: 2D laminar test case

Sensitivities: “bad“ cells



E=1.0

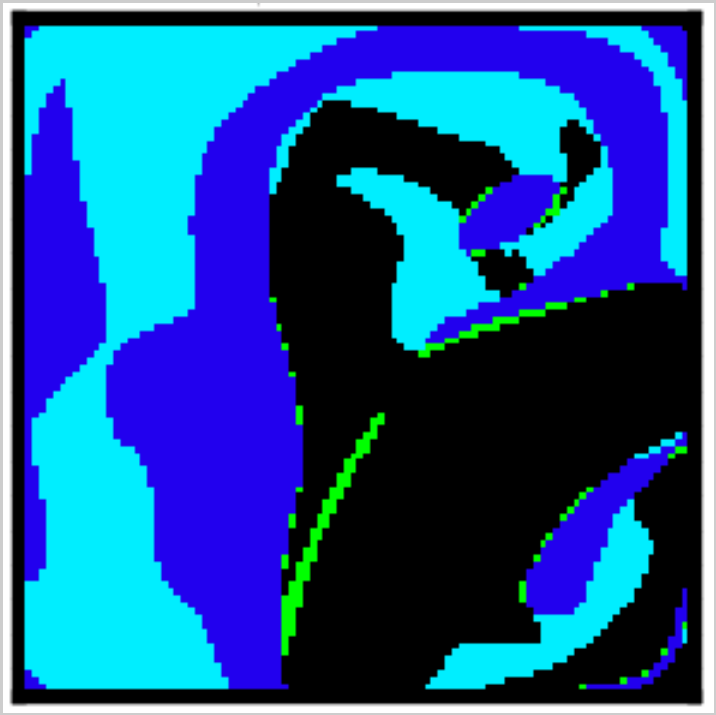
Adjoint approach: 2D laminar test case

Porosity (iter.1)

E=0.88

Adjoint approach: 2D laminar test case

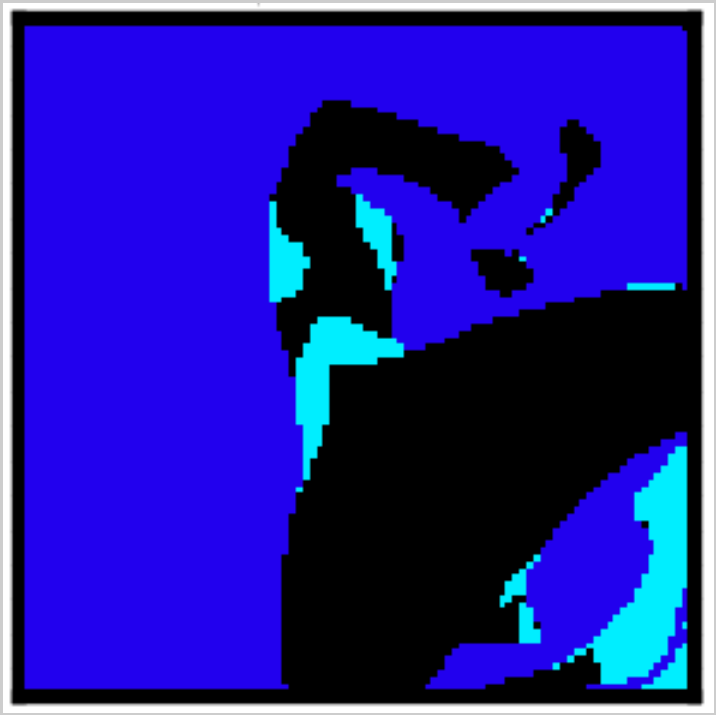
Porosity (iter.2)



E=0.79

Adjoint approach: 2D laminar test case

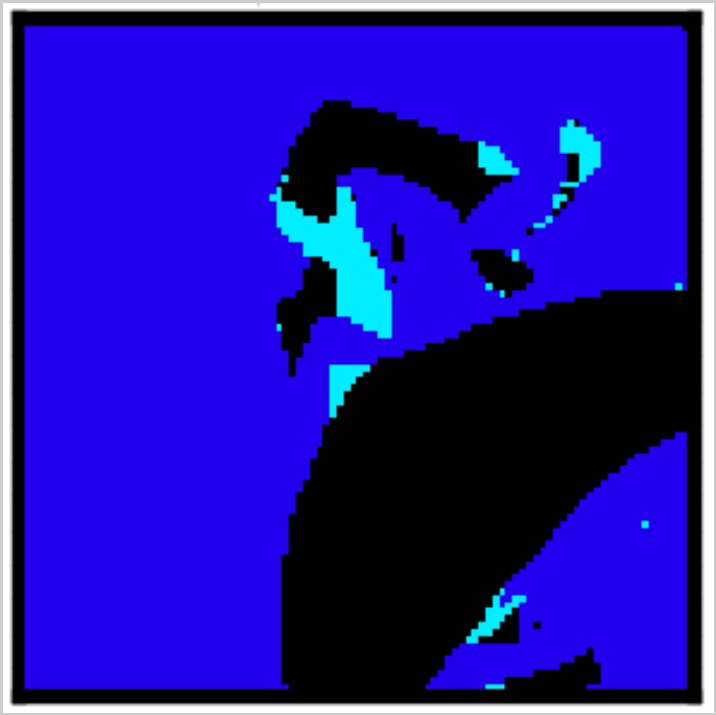
Porosity (iter.3)



E=0.75

Adjoint approach: 2D laminar test case

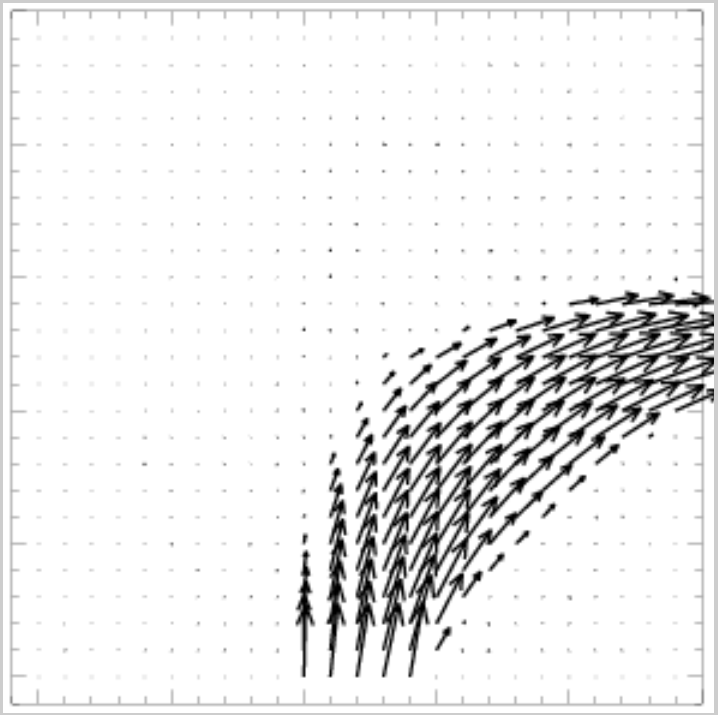
Porosity (iter.4)



E=0.74

Adjoint approach: 2D laminar test case

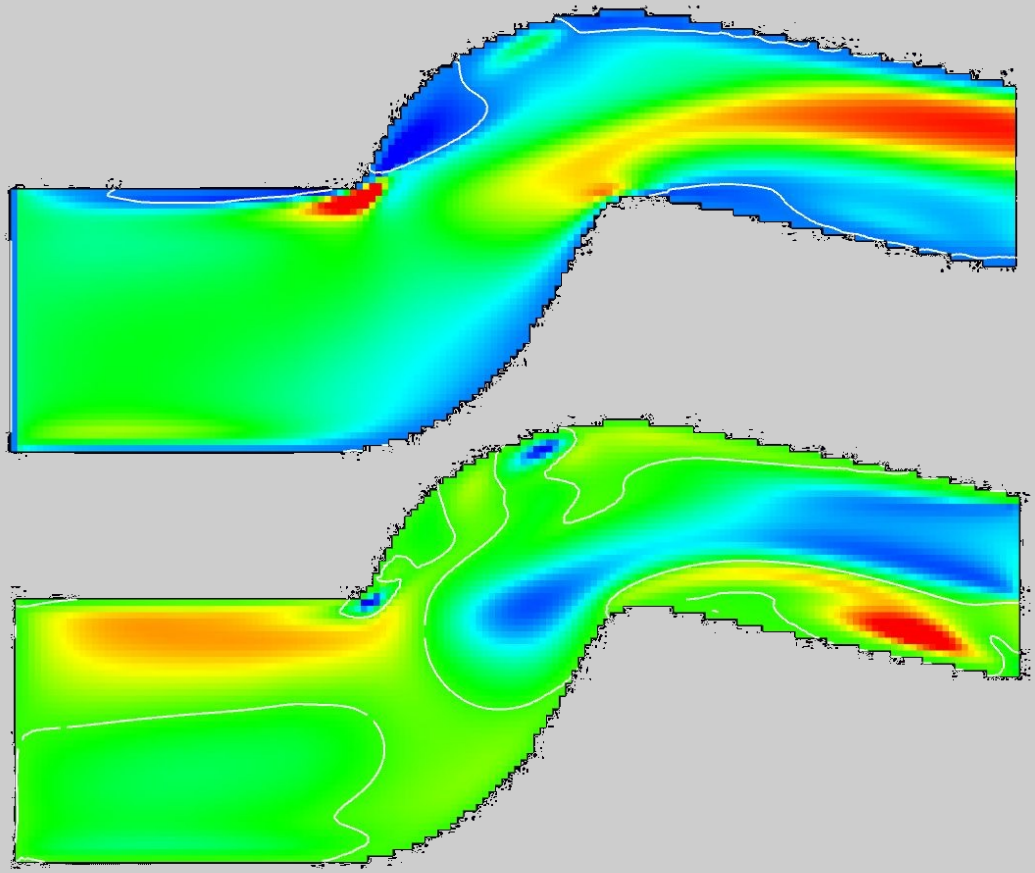
Final velocity



E=0.74

Sensitivity and optimum depend on target function

- Sensitivities for air duct example: good and bad cells

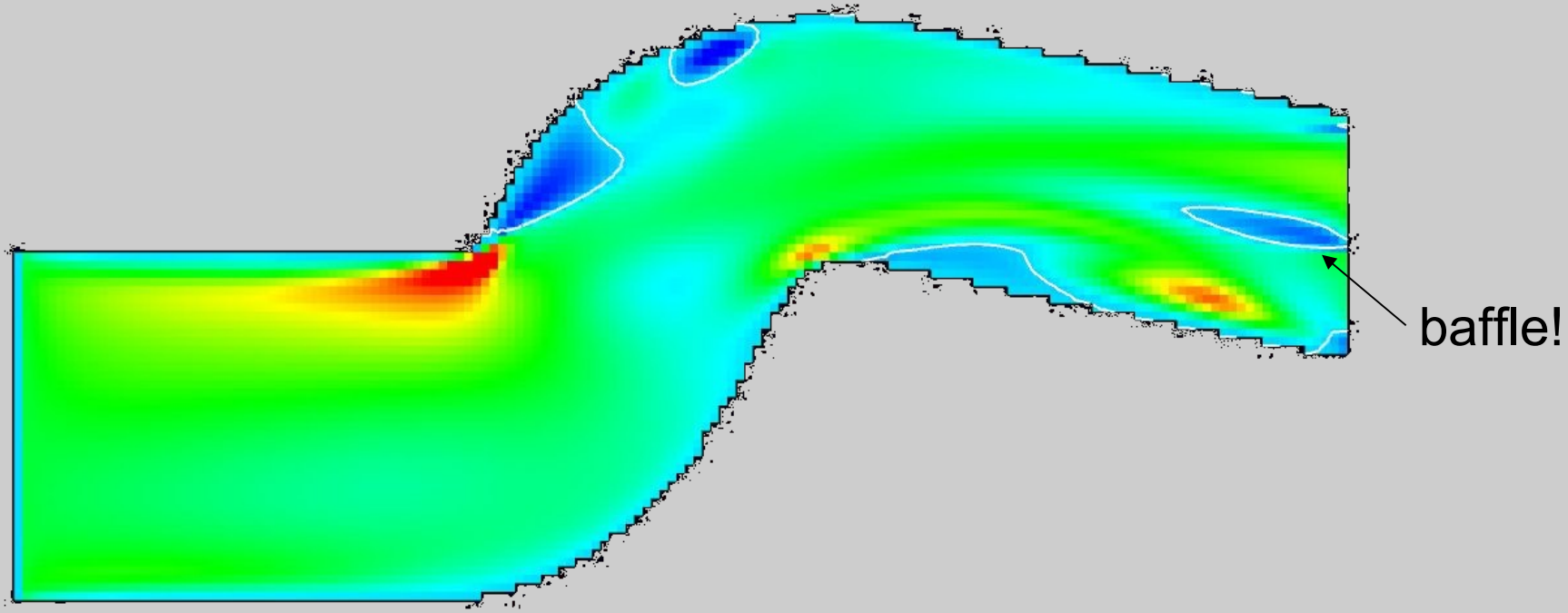


Dissipated power

Flow uniformity

Combined target function

- Sensitivities for air duct example: good and bad cells
- 1*uniformity + 2*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

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

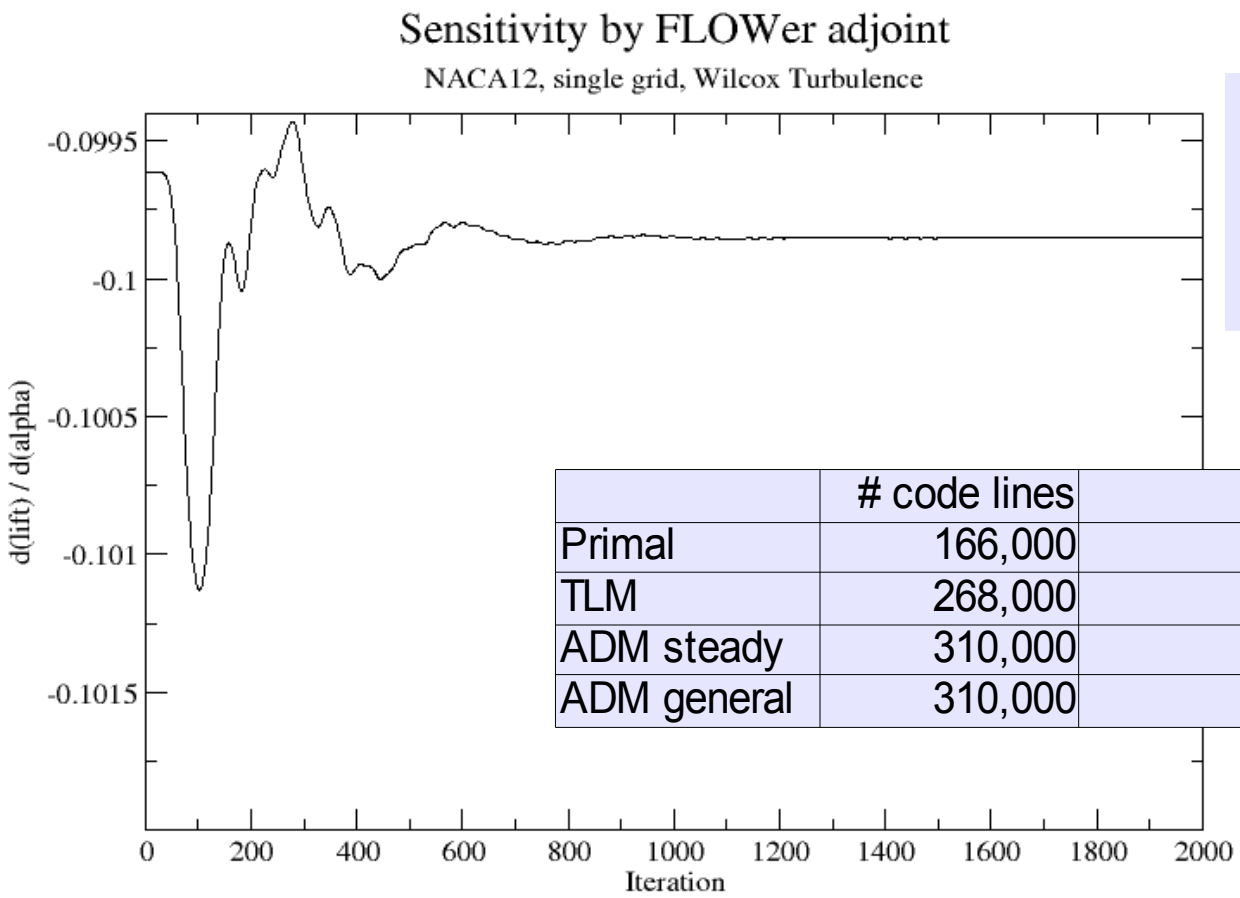
- Lines: total number of Fortran lines *without* comments
- Numbers for TLM and ADM give CPU time for (function + gradient) relative to forward model
- HES format: CPU time for Hessian * n vectors rel. t. forw. model/ n
- 2 (3) level checkpointing costs 1 (2) additional model run(s)

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:

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

AD of FLOWer within MEGADESIGN



Joint work with DLR (Eisfeld, Gauger, Kroll, Raddatz)

in progress ...

	# code lines	memory	CPU (solve+grad)	rel acc. vs FD
Primal	166,000	1	1.0	
TLM	268,000	~2	~3	1E-08
ADM steady	310,000	2—3	6—10	1E-05
ADM general	310,000	variable	<10	1E-08

FastOpt

DLR