Simultaneous Pseudo-Timestepping Methods for Aerodynamic Shape Optimization

Volker Schulz

TEAM: S.B. Hazra (U Trier)
I. Gherman (U Trier)
N. Gauger (DLR)
J. Brezillon (DLR)

University of

Overview

- Pseudo-timestepping gradient methods
- One-shot strategies
- Choosing a proper design Hessian
- Unconstrained numerical results in 2D
- How to deal with constraints?
- Numerical results in 2D and 3D

Research goal

START: Euler flow (Flower)

based on pseudo timestepping (multigrid)

adjoint pseudo timestepping solver

Goal: ***one-shot algorithm for shape optimization***

Pseudo-timestepping as iterative method

$$Ax = b$$

$$\frac{d}{dt}x(t) = b - Ax(t)$$

$$x^{m+1} = x^m + \lambda\left(b - Ax^m\right)$$

$$\frac{d}{dt}x(t) = W\left(b - Ax(t)\right)$$

$$x^{m+1} = x^m + \lambda W\left(b - Ax^m\right)$$

W is called a preconditioner

steepest descent ps-t.

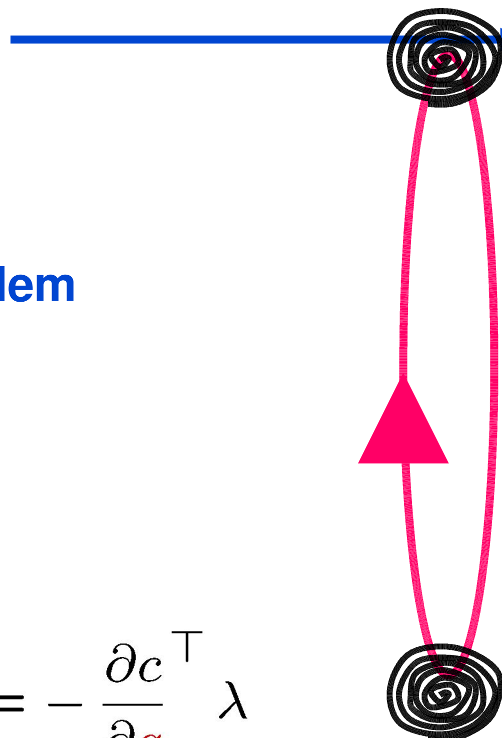
Euler flow

$$c(u) = 0$$

time stepping

$$\dot{u} = -c(u)$$

Flower



optimization problem

$$\min_{q} f(u(q))$$

gradient method

$$q^{k+1} = q^k - \nabla_q f(u(q^k))$$

where

$$\nabla_q f(u(q^k)) = -\frac{\partial c}{\partial q}^\top \lambda$$

with λ from

$$\dot{\lambda} = -\nabla_u \left[f(u, q^k) - \lambda^\top c(u, q^k) \right]$$

Adj. time stepping solver

Volker Schulz

University of

One-shot approach

- How to break up the nested iteration loop?
- *Idea:* one pseudo-time loop for all variables

$$\min_{(u,q)} f(u,q) \quad \text{s.t. } c(u,q) = 0 \quad \longrightarrow \quad \begin{aligned} \nabla_u \left[f(u,q) - \lambda^\top c(u,q) \right] &= 0 \\ \nabla_q \left[f(u,q) - \lambda^\top c(u,q) \right] &= 0 \\ c(u,q) &= 0 \end{aligned}$$

One-shot pseudo-time loop

$$\begin{pmatrix} \dot{\lambda} \\ \dot{q} \\ \dot{u} \end{pmatrix} = -\mathbf{W} \begin{pmatrix} \nabla_u \left[f(u,q) - \lambda^\top c(u,q) \right] \\ \nabla_q \left[f(u,q) - \lambda^\top c(u,q) \right] \\ c(u,q) \end{pmatrix}$$

- *W* = *I* leads to slow convergence, if at all
- *W* should improve convergence
- *W* should be „implementation-friendly“
- Borrow *W* from approximate reduced SQP methods

The essence of reduced SQP techniques:

Instead of incrementing by a full SQP method

$$\begin{bmatrix} H_{uu} & H_{uq} & C_u^* \\ H_{qu} & H_{qq} & C_q^* \\ C_u & C_q & 0 \end{bmatrix} \begin{pmatrix} \Delta u \\ \Delta q \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} -\nabla_u \mathcal{L} \\ -\nabla_q \mathcal{L} \\ -c(u,q) \end{pmatrix}$$

compute increments from

$$\begin{bmatrix} 0 & 0 & C_u^* \\ 0 & B & C_q^* \\ C_u & C_q & 0 \end{bmatrix} \begin{pmatrix} \Delta u \\ \Delta q \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} -\nabla_u \mathcal{L} \\ -\nabla_q \mathcal{L} \\ -c(u,q) \end{pmatrix}$$

… in pseudo-time-stepping formulation:

$$\begin{pmatrix} \dot{u} \\ \dot{q} \\ \dot{\lambda} \end{pmatrix} = \begin{bmatrix} 0 & 0 & C_u^* \\ 0 & B & C_q^* \\ C_u & C_q & 0 \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_u \mathcal{L} \\ -\nabla_q \mathcal{L} \\ -c(u,q) \end{pmatrix}$$

Use known preconditioners ***A*** for states and adjoints

$$\begin{pmatrix} \dot{u} \\ \dot{q} \\ \dot{\lambda} \end{pmatrix} = \begin{bmatrix} 0 & 0 & A_u^* \\ 0 & B & C_q^* \\ A_u & C_q & 0 \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_u \mathcal{L} \\ -\nabla_q \mathcal{L} \\ -c(u,q) \end{pmatrix}$$

Appropriate *B*?

Options:

- Exact reduced Hessian
- „wrong“ reduced Hessian constructed by use of the state preconditioner

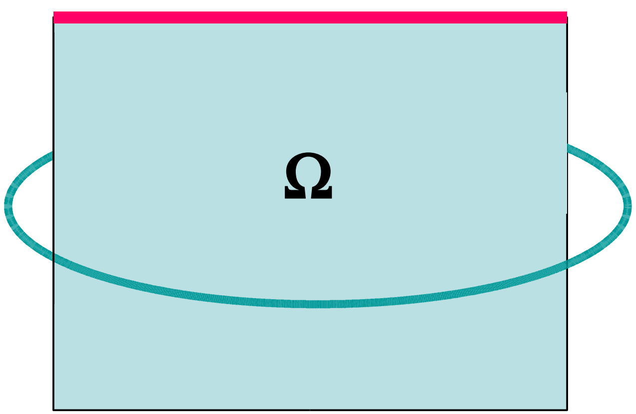
  (cf. Bank/Welfert/Yserentant: A class of iterative methods for solving saddle point problems, Numer. Math. 56, 645-666, 1990)

- Griewank‘s *H(-1)*

Model problem

$$\min_{u,q} \int_{y=1} \left(\frac{\partial u}{\partial \eta} - g(x) \right)^2 dx + \sigma \|q\|_{H^1}^2$$

$$\text{s.t.} \quad \begin{aligned} -\triangle u &= 0 \text{ in } \Omega \\ u &= q(x) \text{ on } y = 1 \\ u &= u_0 \text{ on } y = 0 \end{aligned}$$



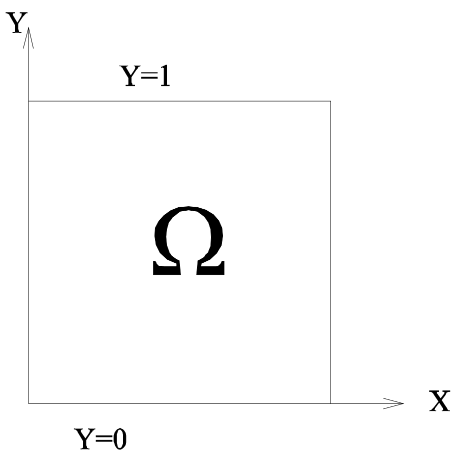
Elliptic PDE to be solved by a pseudo-timestepping method with RK4

Necessary condition $\Rightarrow$

$$-\Delta\lambda = 0 \quad \text{in } \Omega$$

(Costate) $$\lambda + 2\left(\frac{\partial u}{\partial \eta} - g(x)\right) = 0 \quad \text{on } y = 1$$

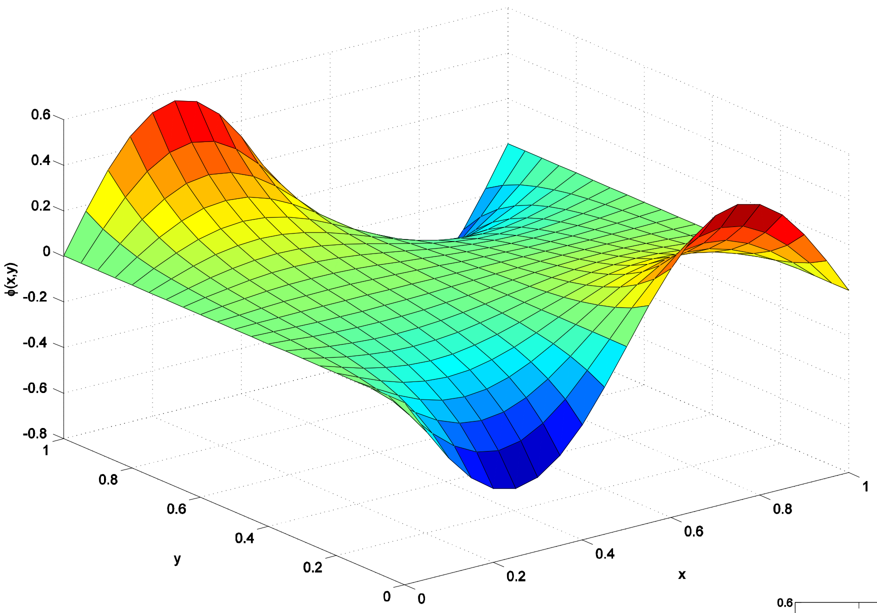
$$\lambda = 0 \quad \text{on } y = 0$$

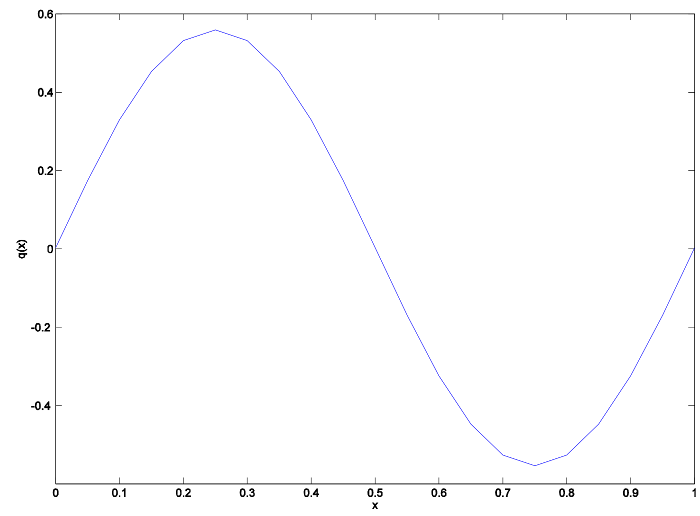


and

(Design) $$2\sigma\left(I - \Delta\right) q - \frac{\partial \lambda}{\partial \eta} = 0 \quad \text{on } y = 1.$$

Solution


0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
ϕ(x,y)
1
0.8
0.6
0.4
0.2
0
y
0
0.2
0.4
0.6
0.8
1
x


0.6
0.4
0.2
0
-0.2
-0.4
q(x)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x

Pseudo-time embedding (unpreconditioned)

$$\frac{d\phi}{dt} - \Delta\phi = 0 \quad \text{in} \quad \Omega$$

$$\frac{d\phi}{dt} + \phi - q(x) = 0 \quad \text{on} \quad y = 1$$

$$\frac{d\phi}{dt} + \phi - \phi_0 = 0 \quad \text{on} \quad y = 0$$

$$\frac{d\lambda}{dt} - \Delta\lambda = 0 \quad \text{in} \quad \Omega$$

$$\frac{d\lambda}{dt} + \lambda + 2\left(\frac{\partial\phi}{\partial\eta} - g(x)\right) = 0 \quad \text{on} \quad y = 1$$

$$\frac{d\lambda}{dt} + \lambda \qquad\qquad = 0 \quad \text{on} \quad y = 0$$

$$\frac{dq}{dt} + 2\sigma q - \frac{\partial\lambda}{\partial\eta} = 0 \quad \text{on} \quad y = 1 \;.$$

What about the reduced Hessian approximation *B*?

- Interpretation as pseudo-differential operator whose symbol can be investigated analytically

- Application of calculus of variations

Both lead to the result:

$$B = 2(\sigma I - (1+\sigma)\frac{\partial^2}{\partial x^2})$$

A closer look at the RSQP-matrix

Instead of

$$\begin{pmatrix} \dot{u}_i \\ \dot{u}_b \\ \dot{q} \\ \dot{\lambda}_b \\ \dot{\lambda}_i \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & -\triangle \\ 0 & 0 & 0 & I & L_b^* \\ 0 & 0 & B & -I & 0 \\ 0 & I & -I & 0 & 0 \\ -\triangle & L_b & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_{u_i}\mathcal{L} \\ -\nabla_{u_b}\mathcal{L} \\ -\nabla_q\mathcal{L} \\ -c_b \\ -c_i \end{pmatrix}$$

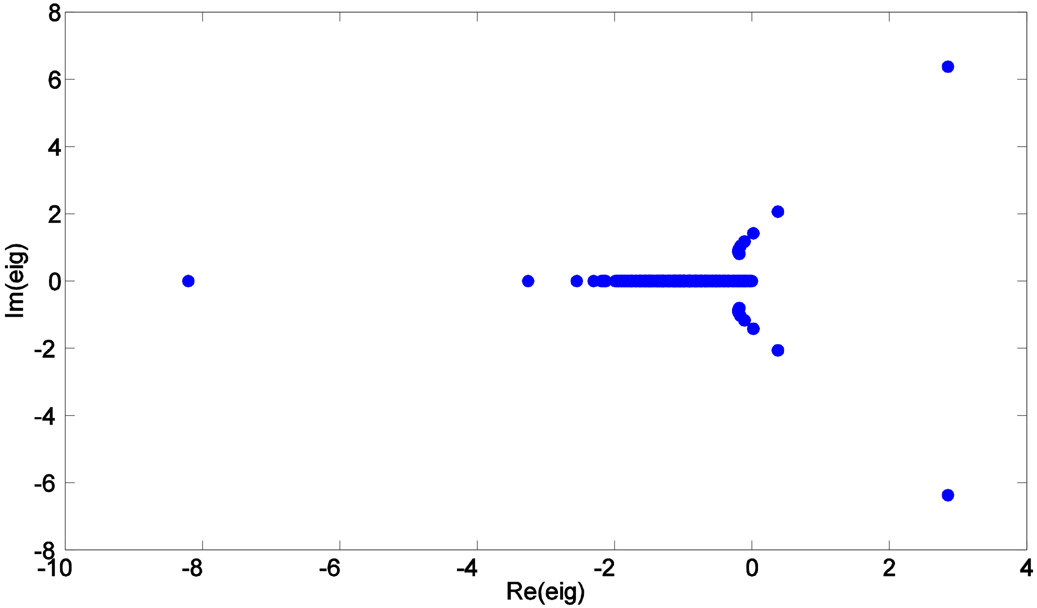
we use

$$\begin{pmatrix} \dot{u}_i \\ \dot{u}_b \\ \dot{q} \\ \dot{\lambda}_b \\ \dot{\lambda}_i \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & -D \\ 0 & 0 & 0 & I & L_b^* \\ 0 & 0 & D_B & -I & 0 \\ 0 & I & -I & 0 & 0 \\ -D & L_b & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_{u_i}\mathcal{L} \\ -\nabla_{u_b}\mathcal{L} \\ -\nabla_q\mathcal{L} \\ -c_b \\ -c_i \end{pmatrix}$$

First Test

- *D* is diagonal of Laplacian (Jacobi prec)
- *B* is exact reduced Hessian
- One-shot pseudo-time integration by classical RKF45

Eigenvalues of resulting dynamical system – <u>no convergence</u>

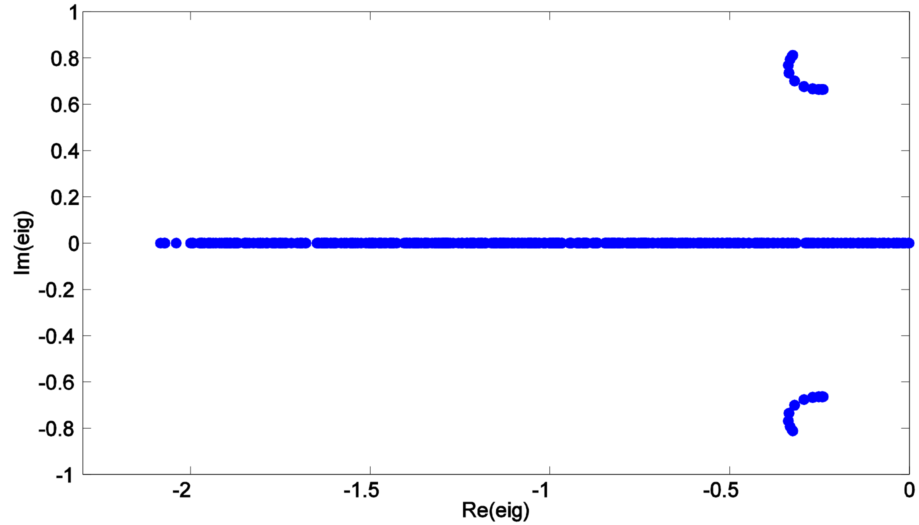


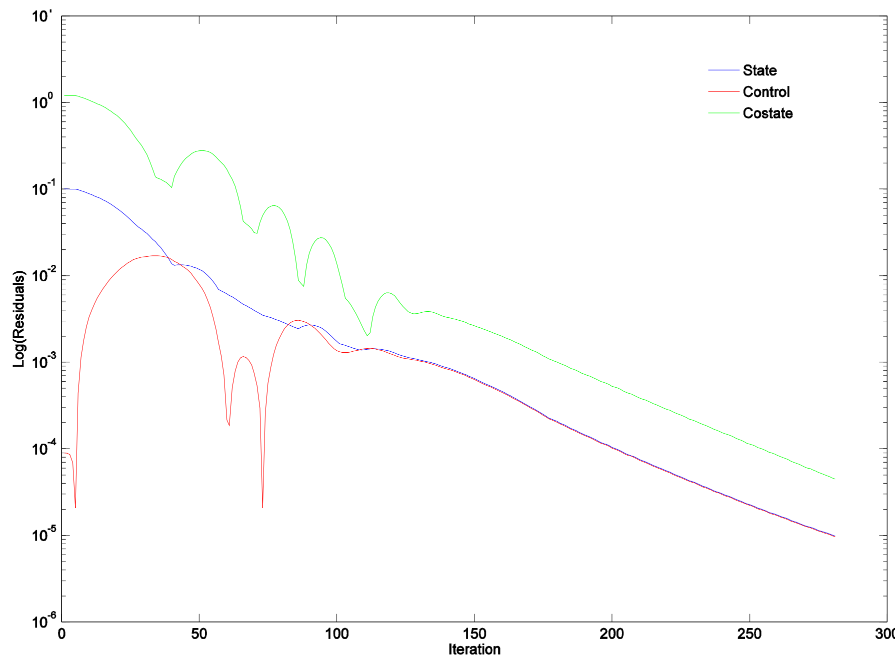
Second Test

- *D* is diagonal of Laplacian (Jacobi prec)
- *B* is „wrong“ reduced Hessian

  (built by the use of *D* instead of Laplacian)

Eigenvalues of resulting dynamical system – <u>convergence</u>



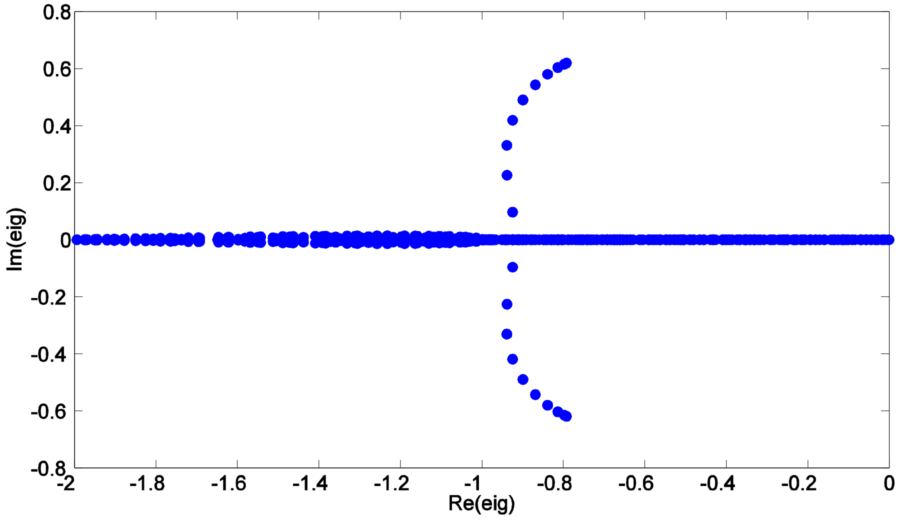


Volker Schulz

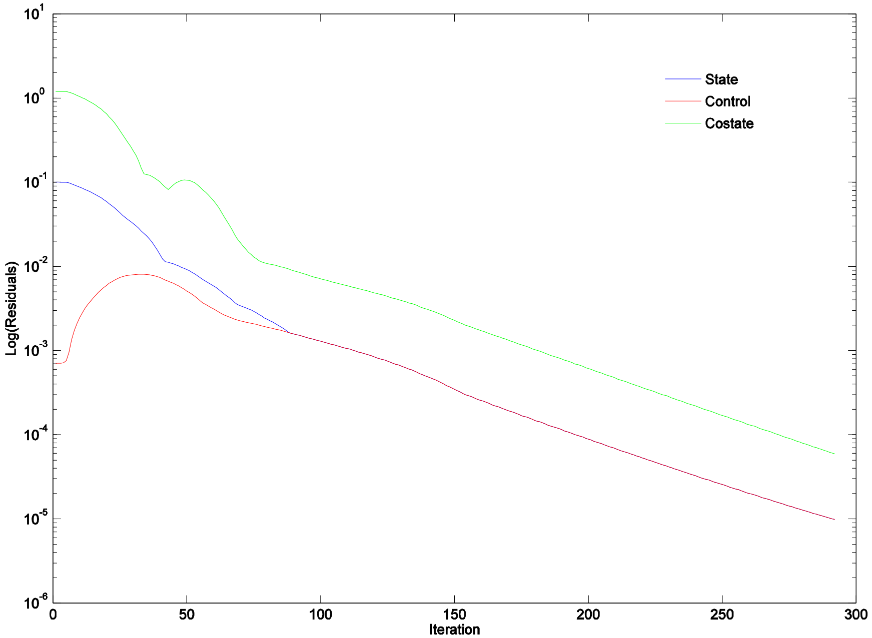
3rd Test

Eigenvalues of resulting dynamical system – <u>convergence</u>

- *D* is diagonal of Laplacian (Jacobi prec)
- *B* is diagonal of exact Hessian



$$\frac{\text{optimization effort}}{\text{state effort}} < 4$$

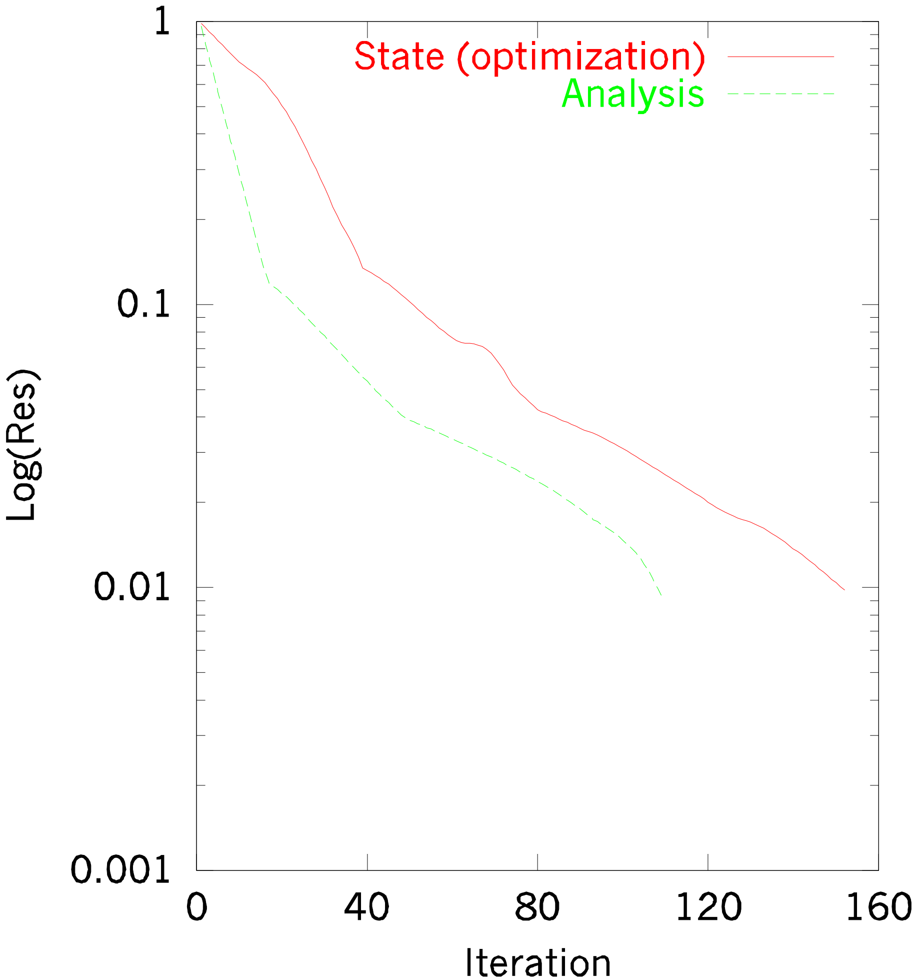


Volker Schulz

consequences

- 4th test with D=Laplacian and exact reduced Hessian yields also convergence but by a tremendous computational effort
- ***We conclude*** that reduced Hessian approximation should be constructed consistent with forward preconditioner
- Too much effort in producing the exact reduced Hessian is wasted.

Numerical result for 20x20 finite difference discretization: overall Runge-Kutta integration



one-shot optimization

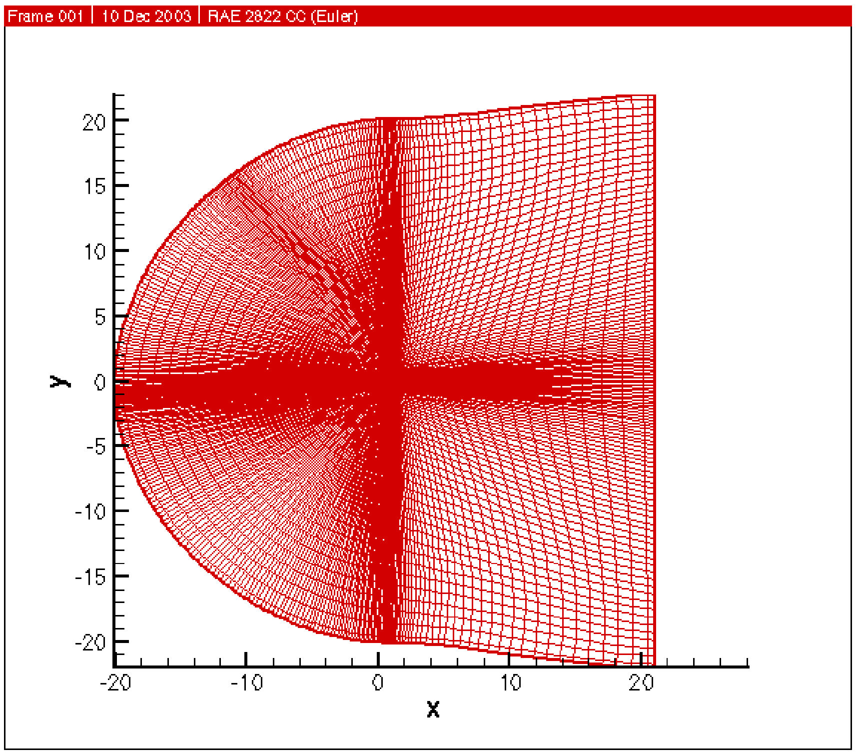
$$\frac{\text{optimization effort}}{\text{state effort}} < 4$$

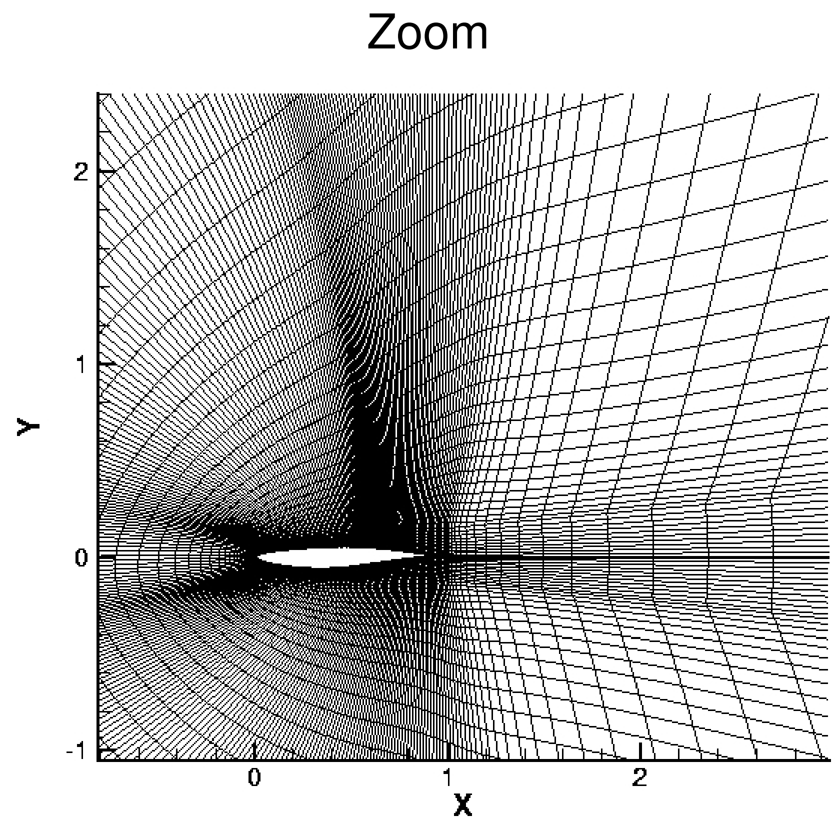
→ *Without preconditioning factor 1000 !*

Volker Schulz

University of

Minimize Drag of an RAE 2822 airfoil by use of *Flower*/DLR within rSQP one shot optimization





Volker Schulz

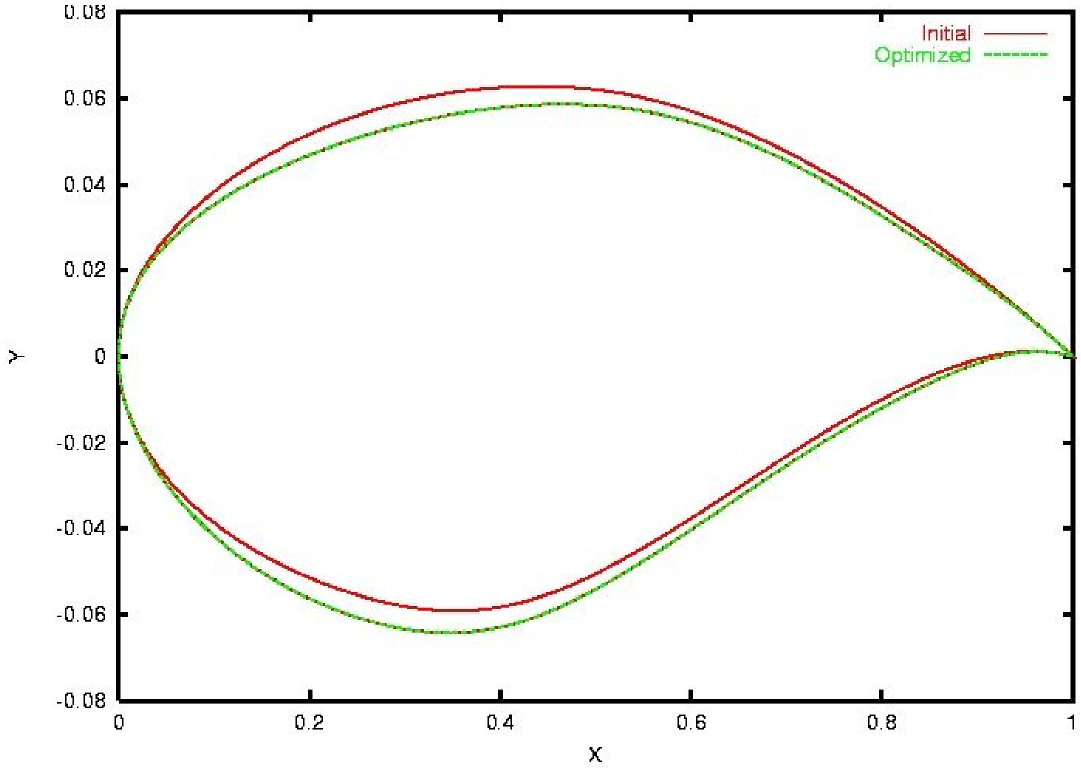
University of

2D-results

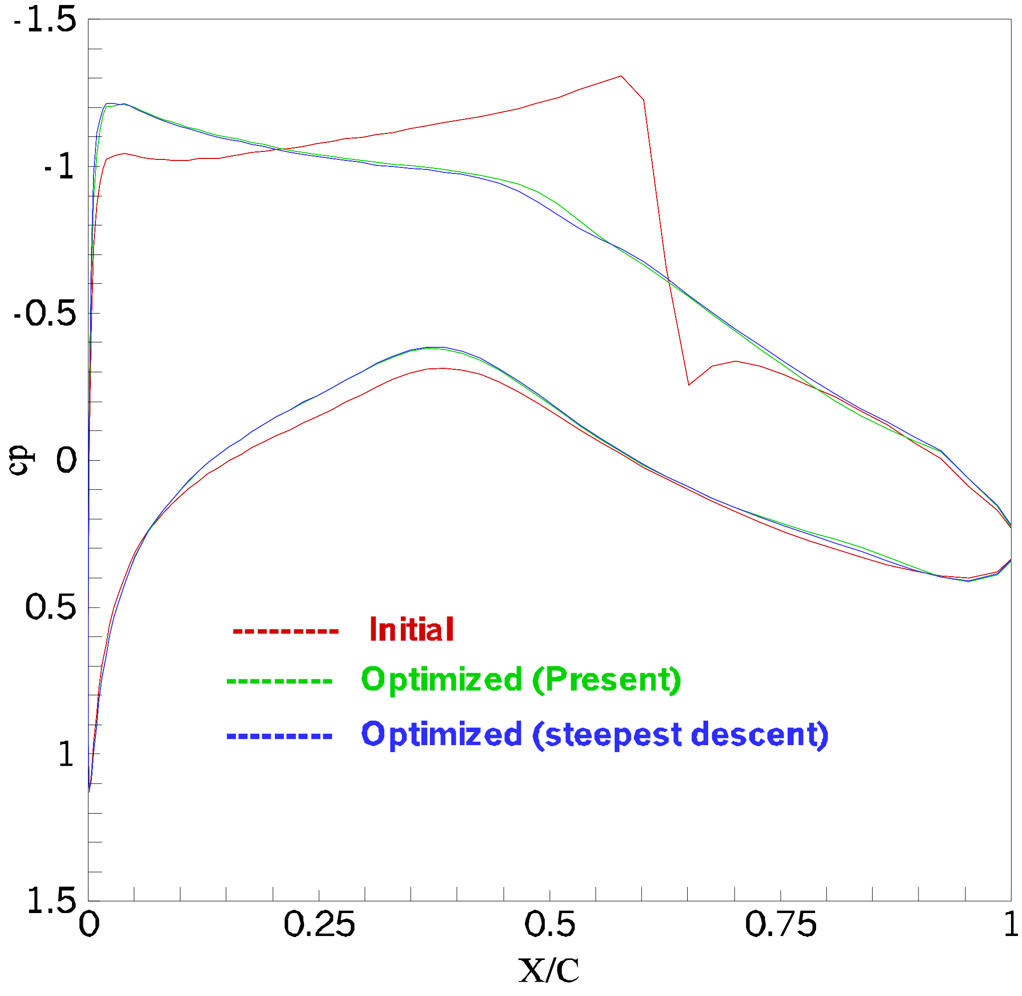
- Euler flow (Code: *Flower*/DLR) with adjoint solver by Gauger
- Minimize drag subject to constant profile thickness: 66% reduction achieved
- Technique employed per iteration:
  - State/adjoint: single RK-steps provided by *Flower*
  - Design: explicit Euler with scalar reduced Hessian approximation of the form: $B \approx \frac{(\gamma^k - \gamma^{k-1})^\top \Delta q^k}{\|\Delta q^k\|^2} \cdot I$

where $\gamma^k = \nabla_q f^k - C_q^\top \lambda^k \approx \text{"reduced gradient"}$

Profile change



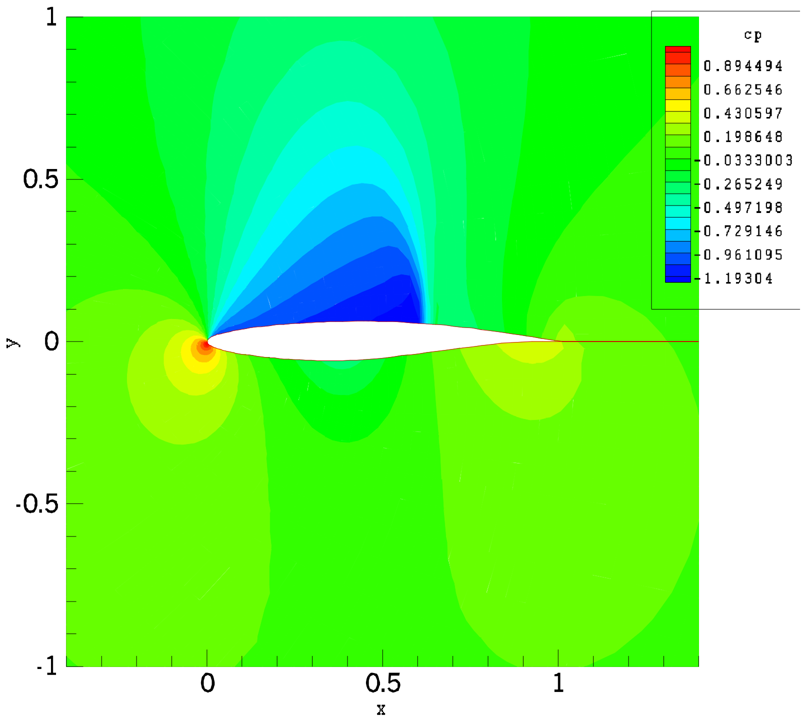
Pressure over profile length



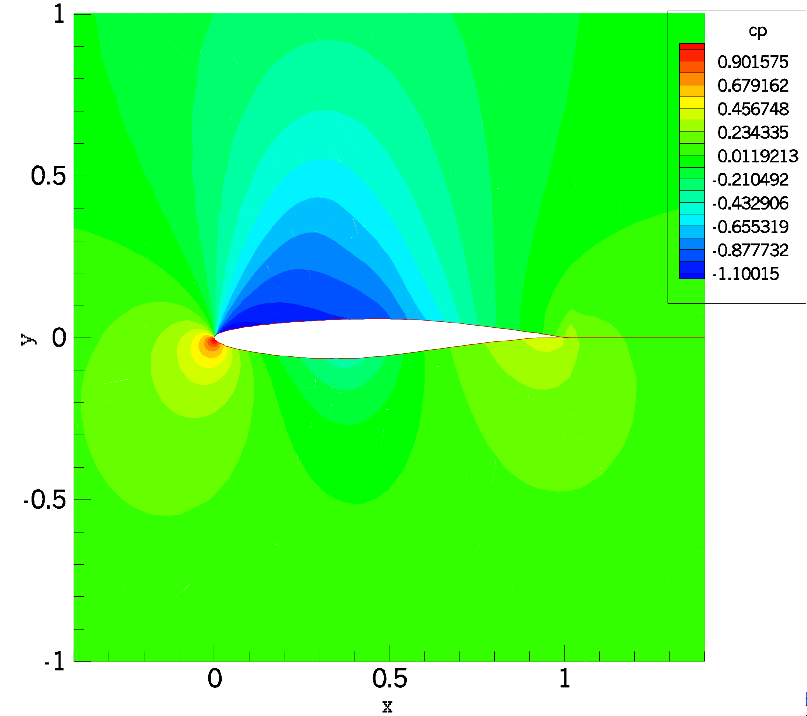
Volker Schulz

Pressure

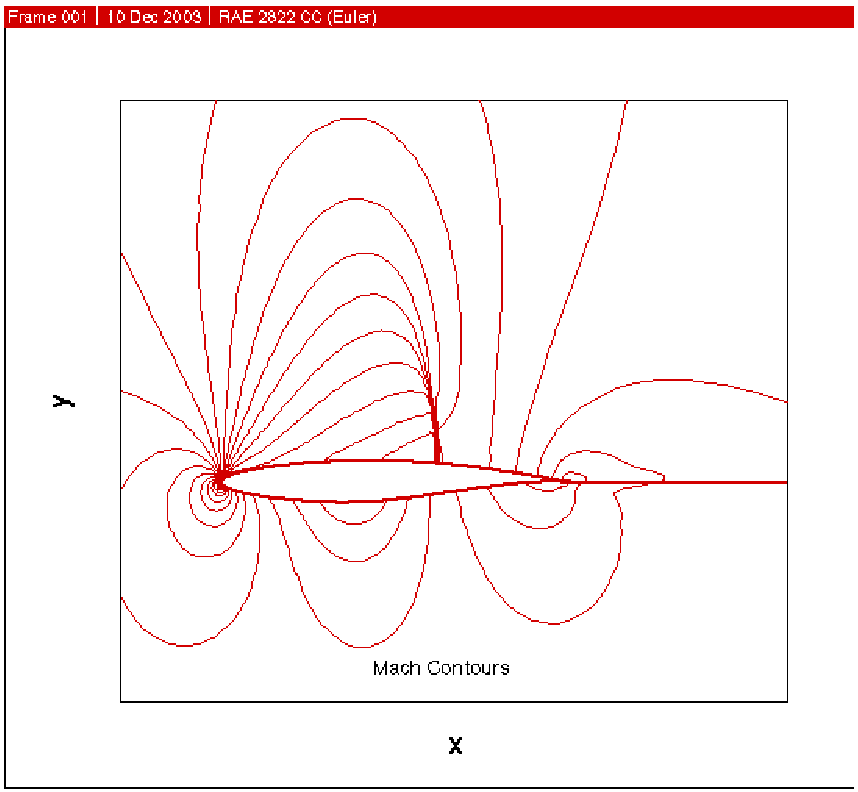
before opt

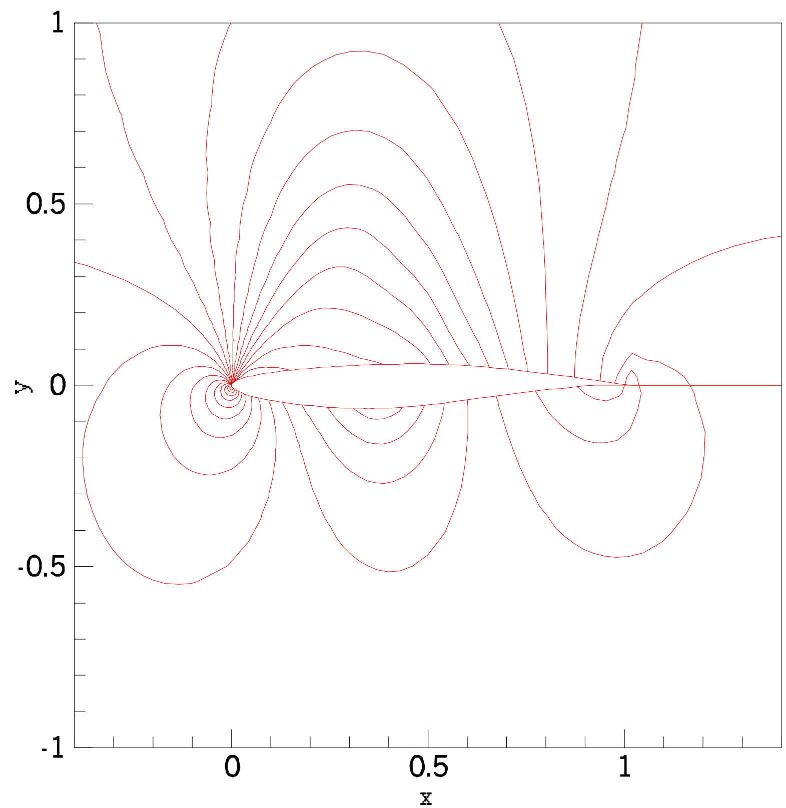


after opt

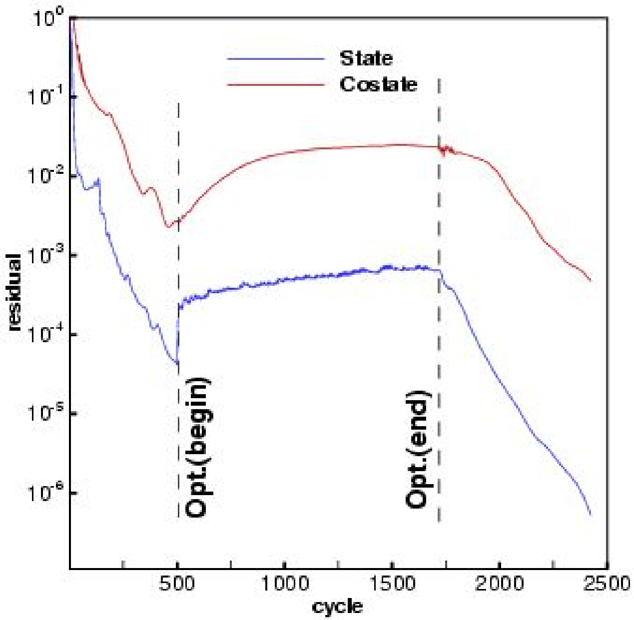


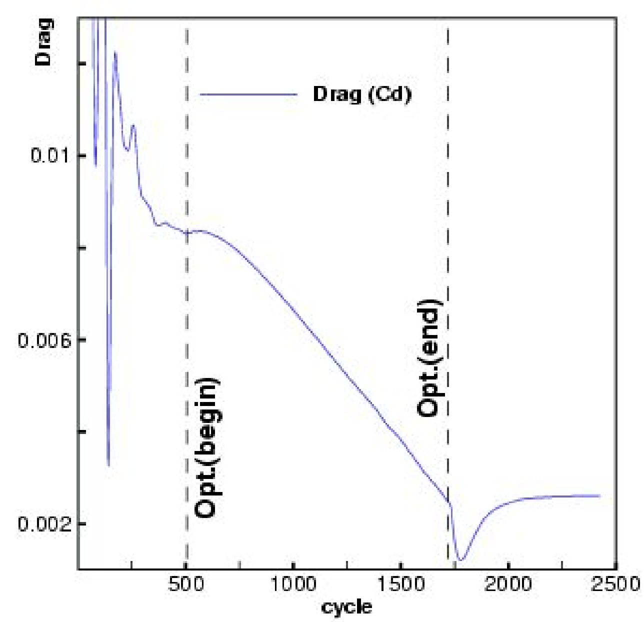
Mach number (velocity)





Convergence history

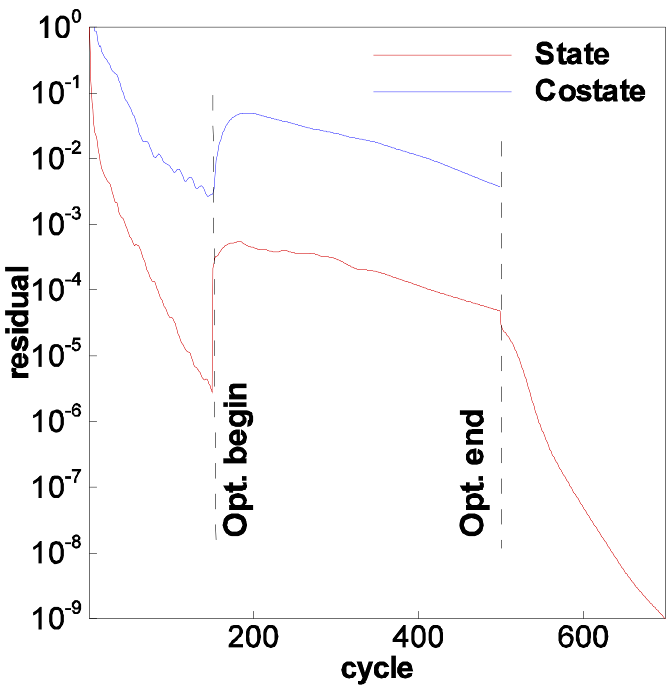


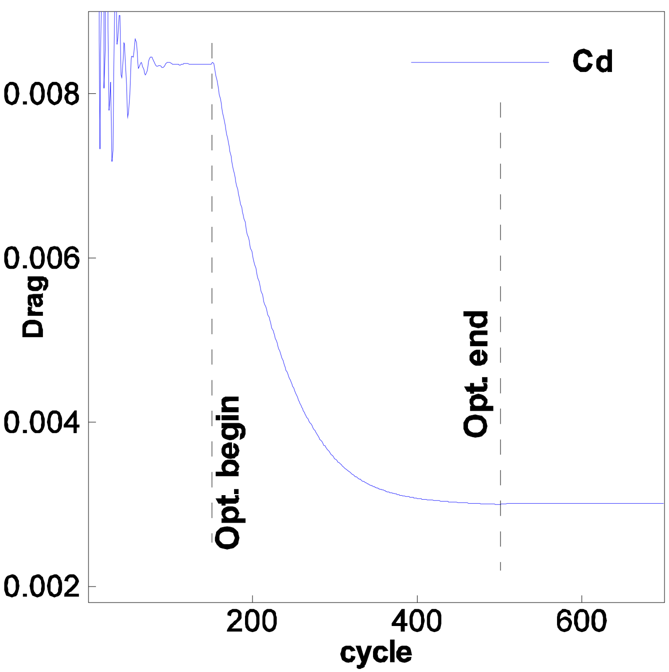


One forward run requires 1500 iterations

$$\Rightarrow \frac{\text{optimization effort}}{\text{state effort}} < 4$$

Multigrid results
(3 grid levels)

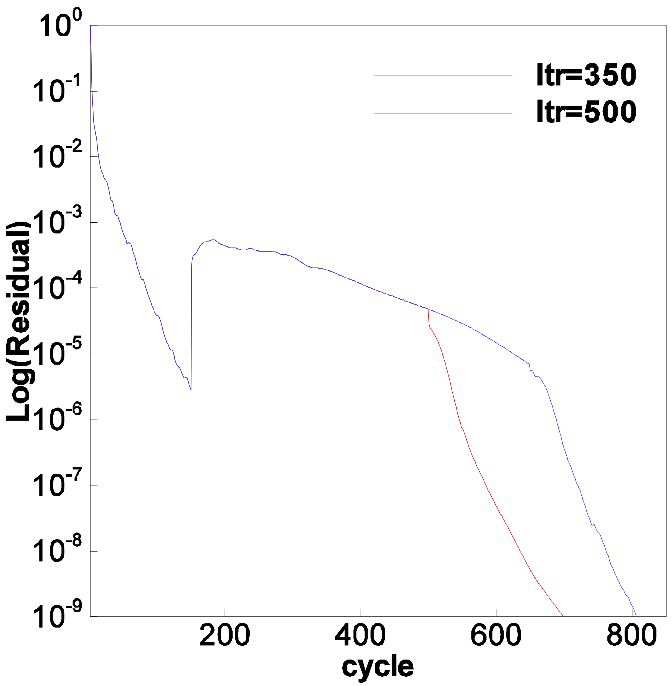


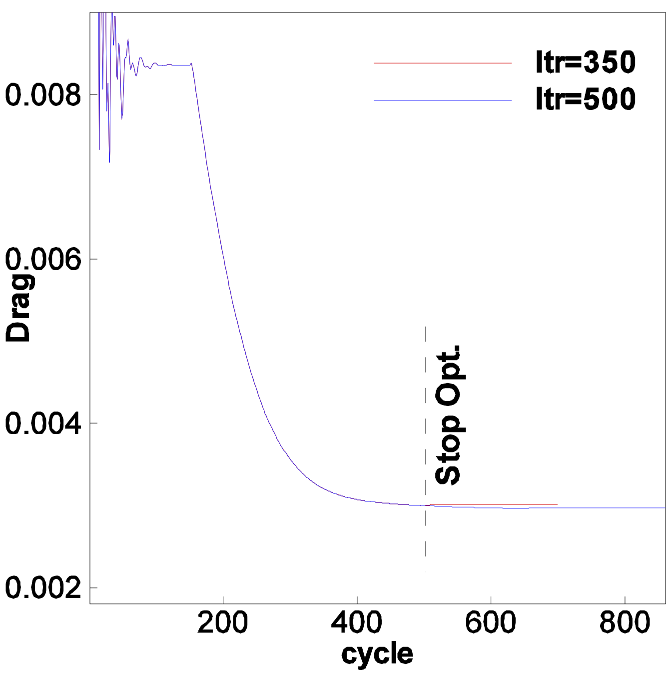


Volker Schulz

University of

Iterating to „infinity"





Volker Schulz

University of

State constraints

Real goal:

minimize **drag**

s.t. **lift >= $\underline{l}$**

where **lift: (u,q) R^1**

depends on the states and can be computed by the solution of yet another adjoint problem!

(no adjustment of angle of incidence)

Algorithmic extension

$$\min_{u,q} f(u,q)$$
$$\text{s.t.} \quad h(u,q) \geq h_0 \quad \in R^1$$
$$c(u,q) = 0, \quad \exists c_u^{-1}$$

Newton-KKT

$$\begin{bmatrix} H_{uu} & H_{uq} & h_u^* & C_u^* \\ H_{qu} & H_{qq} & h_q^* & C_q^* \\ h_u & h_q & 0 & 0 \\ C_u & C_q & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta u \\ \Delta q \\ \Delta \mu \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} -\nabla_u \mathcal{L} \\ -\nabla_q \mathcal{L} \\ -h(u,q) \\ -c(u,q) \end{pmatrix}$$

is substituted by time-evolution

$$\begin{bmatrix} 0 & 0 & 0 & D_u^* \\ 0 & B & \tilde{\gamma}^* & C_q^* \\ 0 & \tilde{\gamma} & 0 & 0 \\ D_u & C_q & 0 & 0 \end{bmatrix} \begin{pmatrix} \dot{u} \\ \dot{q} \\ \dot{\mu} \\ \dot{\lambda} \end{pmatrix} = \begin{pmatrix} -\nabla_u \mathcal{L} \\ -\nabla_q \mathcal{L} \\ -h(u,q) \\ -c(u,q) \end{pmatrix}$$

Where $\tilde{\gamma}$ denotes the current gradient approximation from adjoint lift time-evolution

Volker Schulz

University of

One approximate Newton step for the constraint problem:

$$\begin{pmatrix} 0 & 0 & \left(\frac{\partial h}{\partial w}\right)^\top & A^\top \\ 0 & B & \left(\frac{\partial h}{\partial q}\right)^\top & \left(\frac{\partial c}{\partial q}\right)^\top \\ \frac{\partial h}{\partial w} & \frac{\partial h}{\partial q} & 0 & 0 \\ A & \frac{\partial c}{\partial q} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta q \\ \Delta \mu \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -h \\ -c \end{pmatrix}.$$

„Partially reduced SQP method“

Block-Gauss-reduction to:

$$\begin{pmatrix} B & g_h \\ g_h^\top & 0 \end{pmatrix} \begin{pmatrix} \Delta q \\ \Delta \mu \end{pmatrix} = \begin{pmatrix} -\nabla_q L + \left(\frac{\partial c}{\partial q}\right)^\top A^{-\top} \nabla_w L \\ -h + \frac{\partial h}{\partial w} A^{-1} c \end{pmatrix},$$

with the reduced gradient

$$g_h := \left(\frac{\partial h}{\partial q} - \frac{\partial h}{\partial w} A^{-1} \frac{\partial c}{\partial q} \right)^\top = \begin{bmatrix} -A^{-1} \frac{\partial c}{\partial q} \\ I \end{bmatrix}^\top \begin{bmatrix} \nabla_w h \\ \nabla_q h \end{bmatrix}$$

Pseudo-stationary system:

$$\begin{pmatrix} 0 & 0 & \left(\frac{\partial h}{\partial w}\right)^\top & A^\top \\ 0 & B & \left(\frac{\partial h}{\partial q}\right)^\top & \left(\frac{\partial c}{\partial q}\right)^\top \\ \frac{\partial h}{\partial w} & \frac{\partial h}{\partial q} & 0 & 0 \\ A & \frac{\partial c}{\partial q} & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{w} \\ \dot{q} \\ \dot{\mu} \\ \dot{\lambda} \end{pmatrix} = \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -h \\ -c \end{pmatrix}.$$

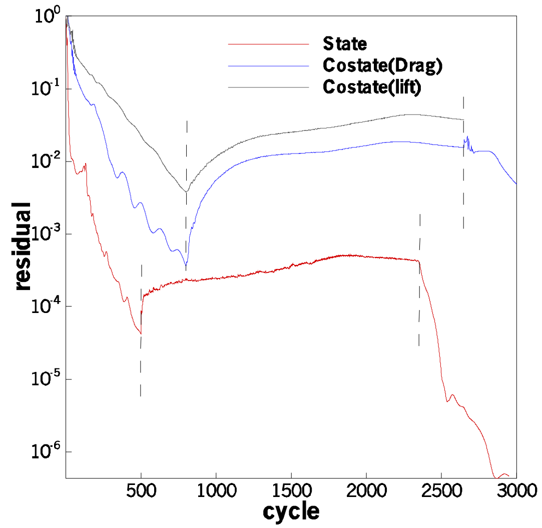
The pseudo-timestepping cycle

1) Perform 1 RK step for state equations
2) Perform 1 RK step for adjoint equations for drag
3) Perform 1 RK step for adjoint equations for lift
4) Solve the QP:

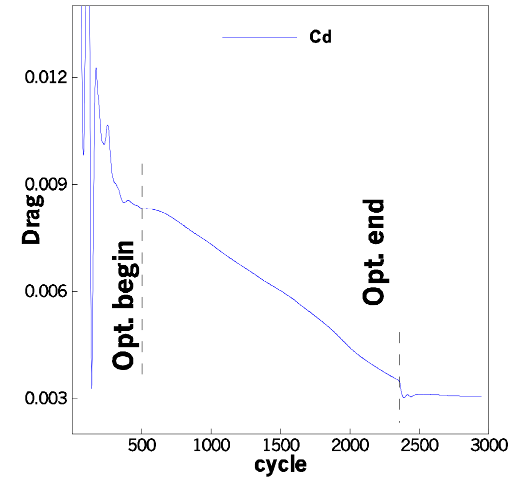
$$\min \; \frac{1}{2}\dot{q}^\top B\dot{q} + g_{lift}^\top \dot{q}$$
$$\text{s.t. } g_h^\top \dot{q} = -h(w,q) + \underline{\frac{\partial h}{\partial w}\dot{w}_{fwd}}$$

5) Perform 1 explicit Euler step for design equation

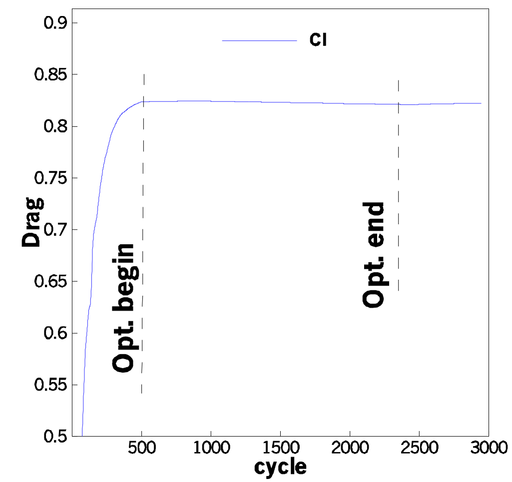
Convergence history



- 62%



- 0.2%

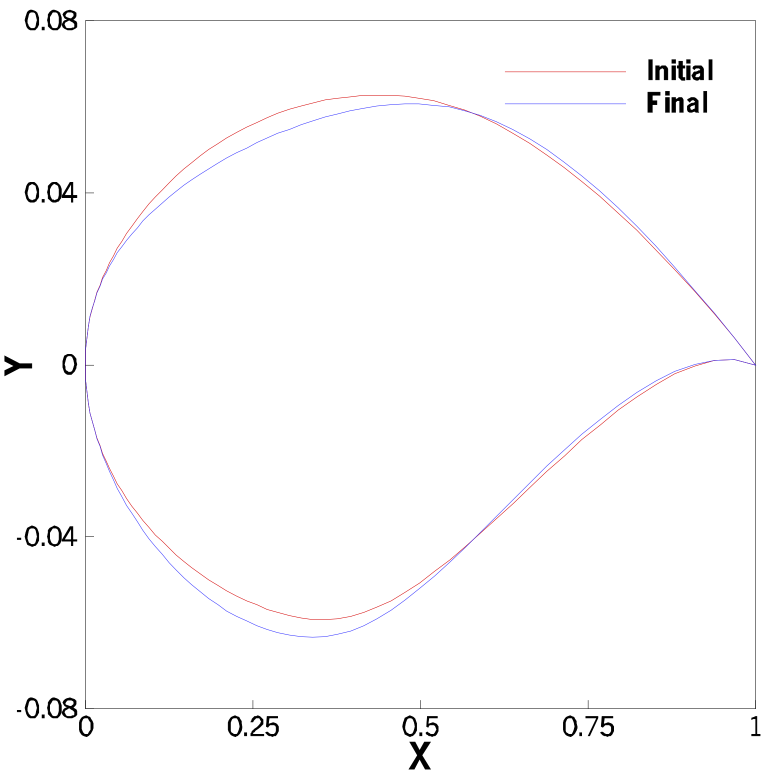


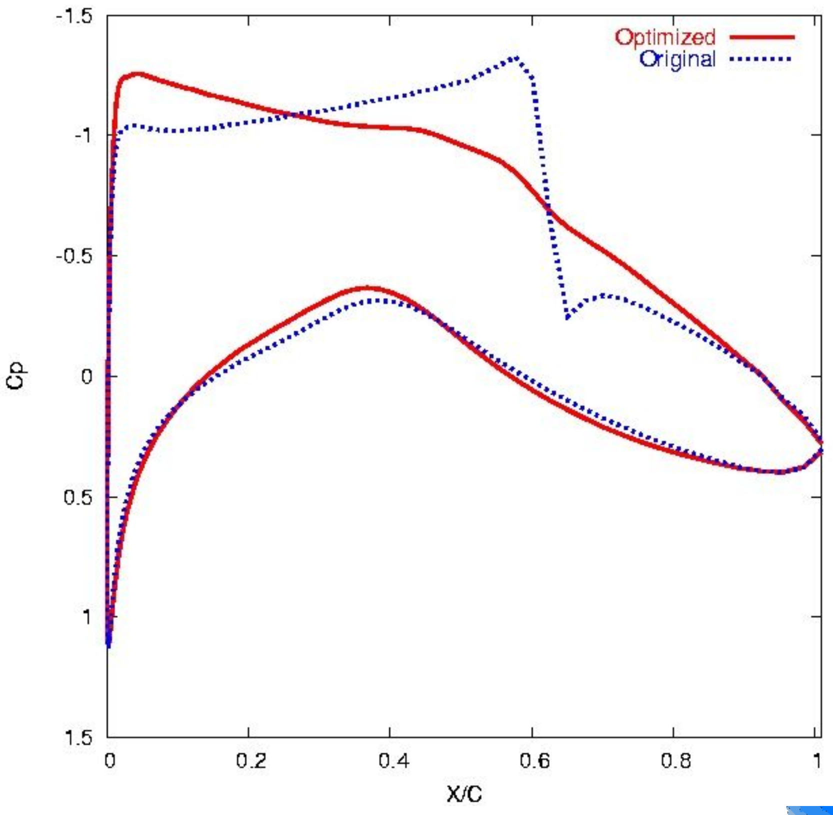
$$\Rightarrow \frac{\text{optimization effort}}{\text{state effort}} < 7$$

Volker Schulz

University of

2D-results for the constrained case





Complexity measurements

- The overall cost of solving the optimization problem is roughly 4 times the cost of the forward problem (without lift constr.)

- Cost with lift constraint. = 7 times the forward problem

Lift & pitching moment constraints

minimize **drag**

s.t. **lift >= c0**

moment >= c1

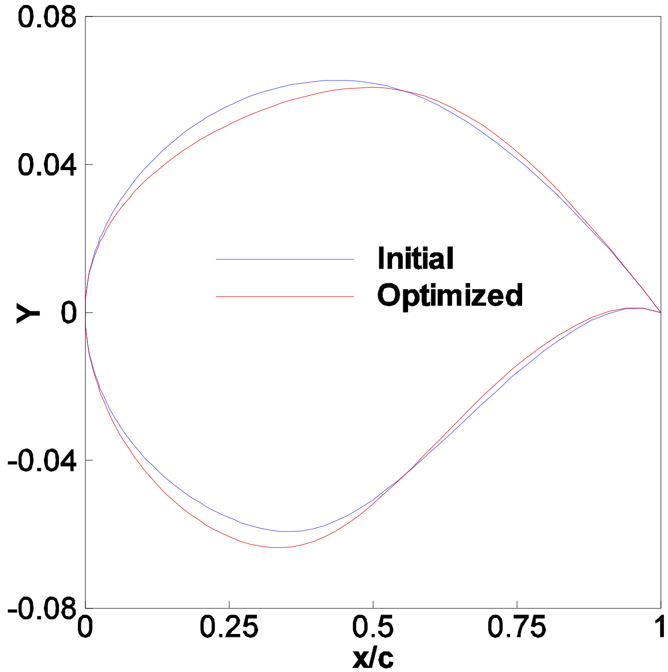
The pseudo-timestepping cycle

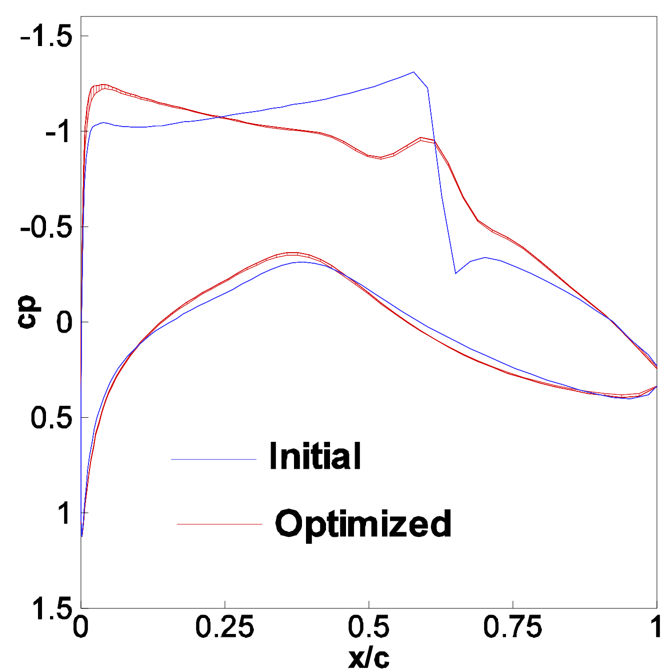
1) Perform 1 RK step for state equations
2) Perform 1 RK step for adjoint equations for drag
3) Perform 1 RK step for adjoint equations for lift
4) Perform 2 RK step for adjoint equations for moment
5) Solve the QP:

$$\min \ \frac{1}{2}\dot{q}^\top B\dot{q} + g_{\text{drag}}^\top \dot{q}$$
$$\text{s.t. } g_h^\top \dot{q} = -h(u,q) + \frac{\partial h}{\partial u}\dot{u}_{fwd}$$
$$g_m^\top \dot{q} = -m(u,q) + \frac{\partial m}{\partial u}\dot{u}_{fwd}$$

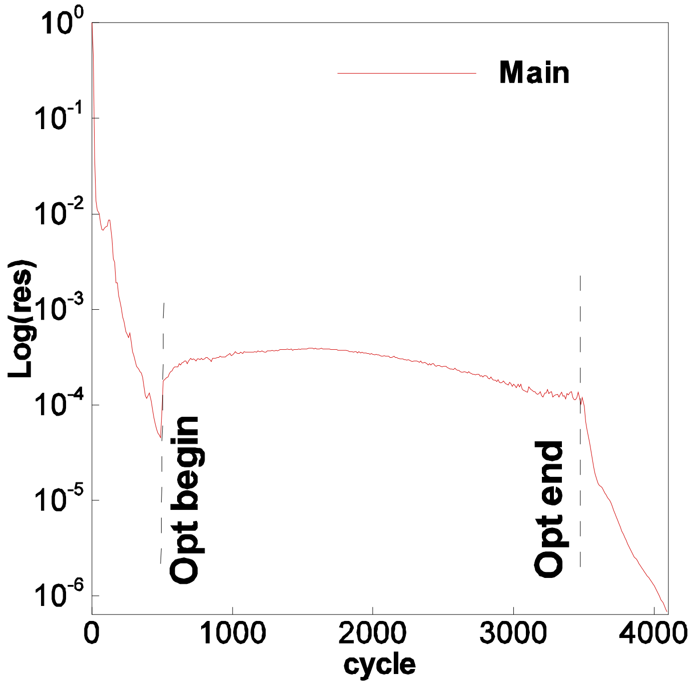
6) Perform 1 explicit Euler step for design equation

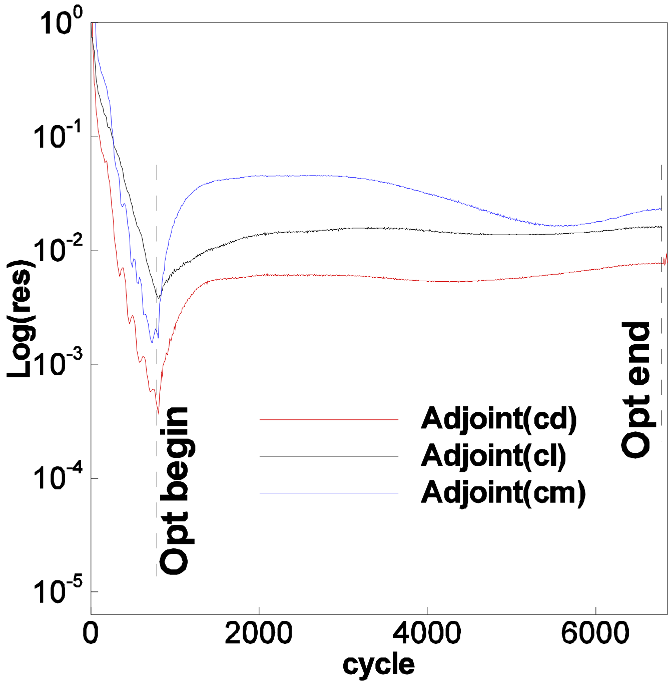
Lift & Pitching moment results





Convergence history



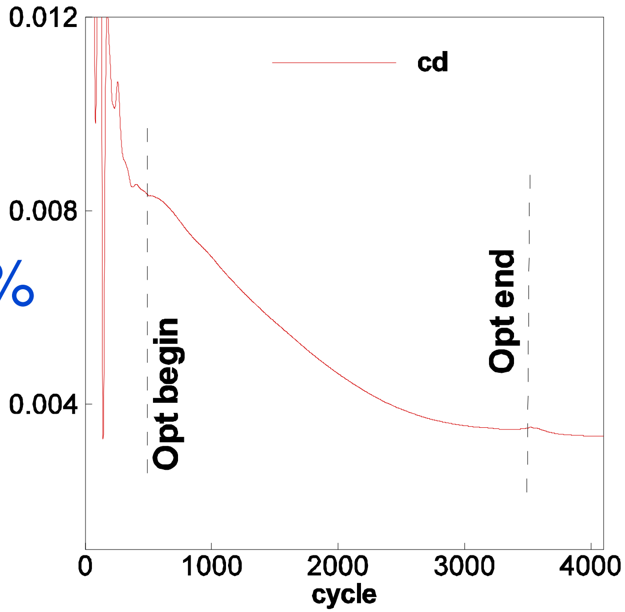


Volker Schulz

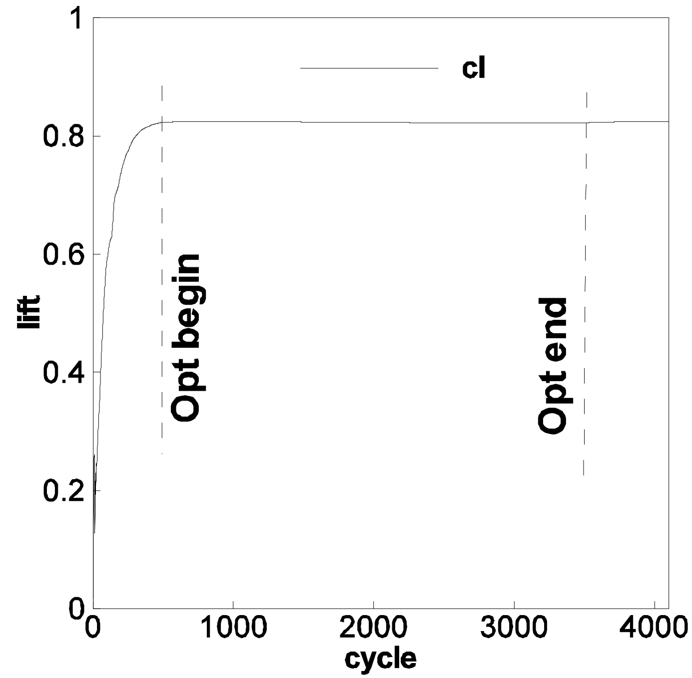
University of

… continued

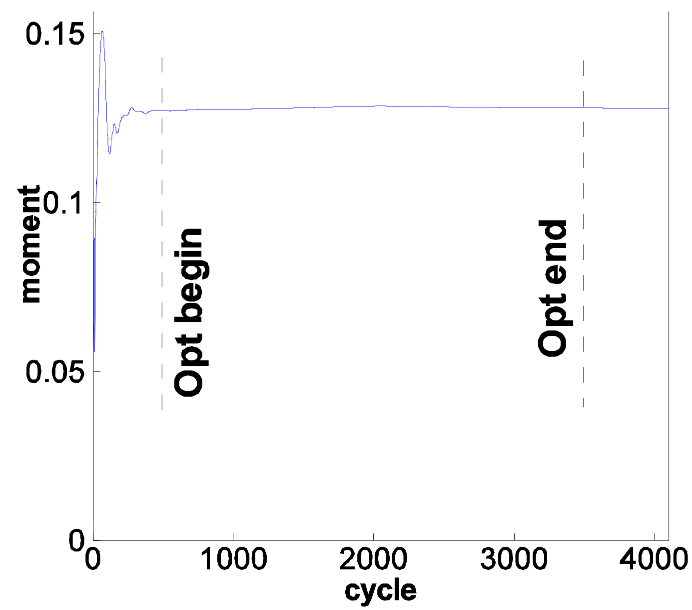
- 60%



+ 0.1%



+ 0.4%



$$\Rightarrow \frac{\text{optimization effort}}{\text{state effort}} < 10$$

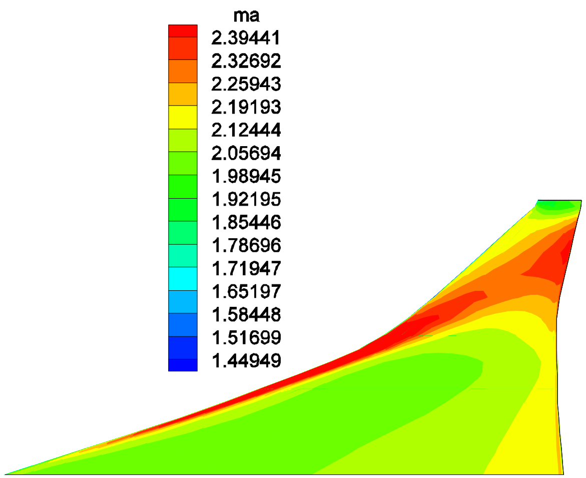
3D results for SCT wing

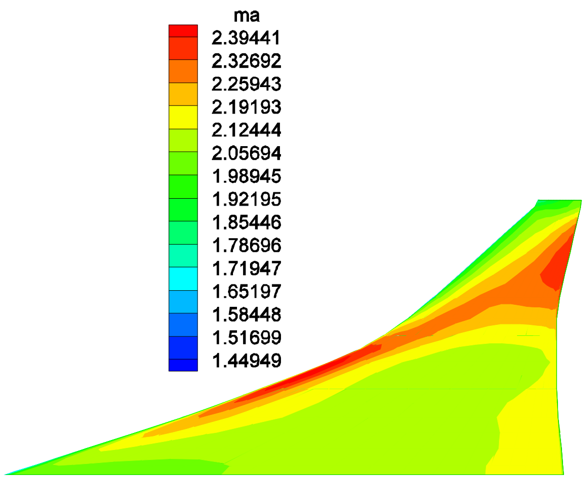
- *S*upersonic *c*ommercial *t*ransport aircraft
- Minimize drag subject to constant lift
- Drag reduced by 12.65%
- Grid: 97 x 17 x 25 = 42 225 grid nodes
- 122 geometry parameters: thickness, camberline, twist, additional DOF: angle of attack

$$\frac{\text{optimization effort}}{\text{state effort}} < 6$$

(2 initial iterations have to be spent to compute approximation of sensitivity lift versus drag)

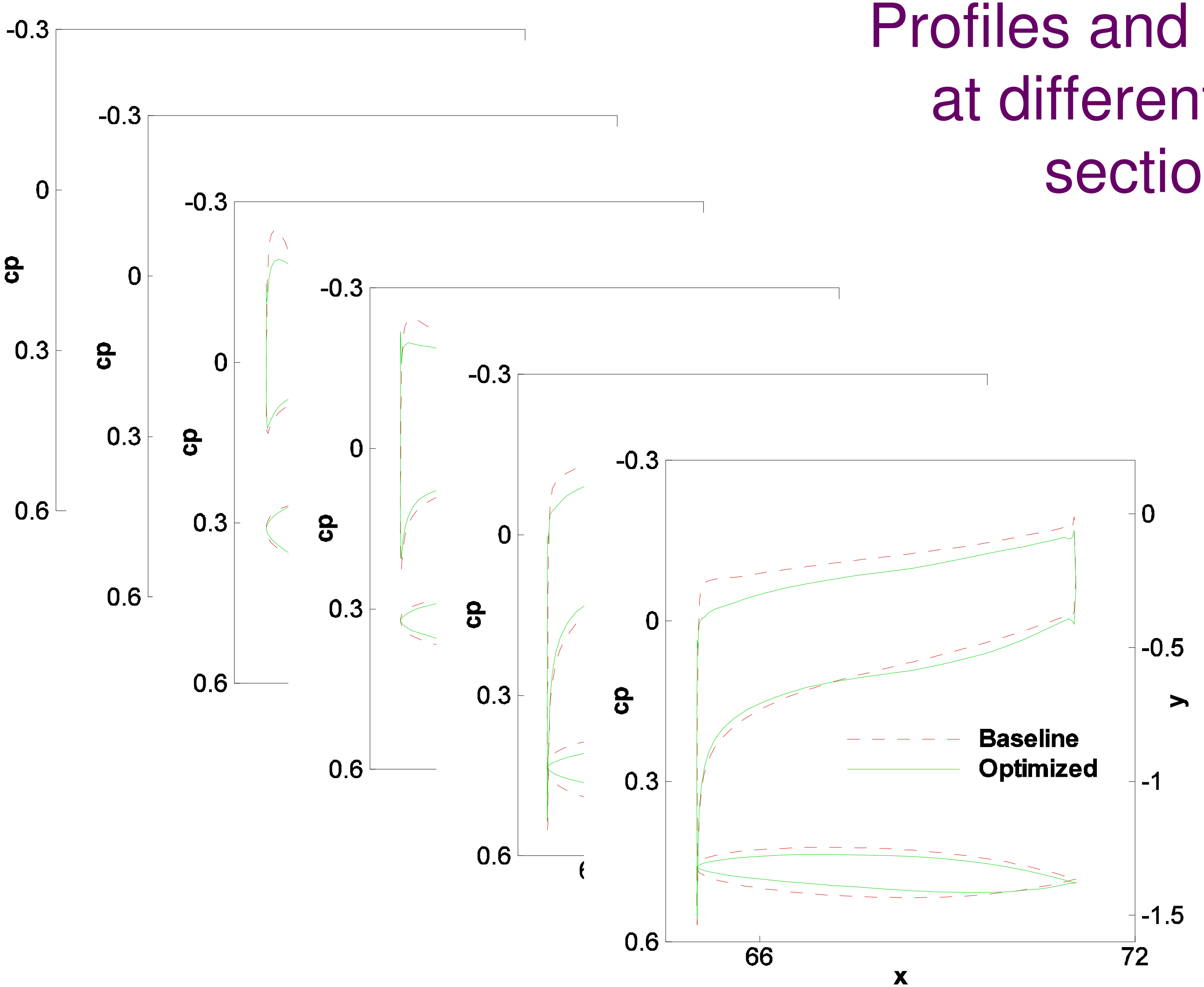
Base velocities versus optimal solution



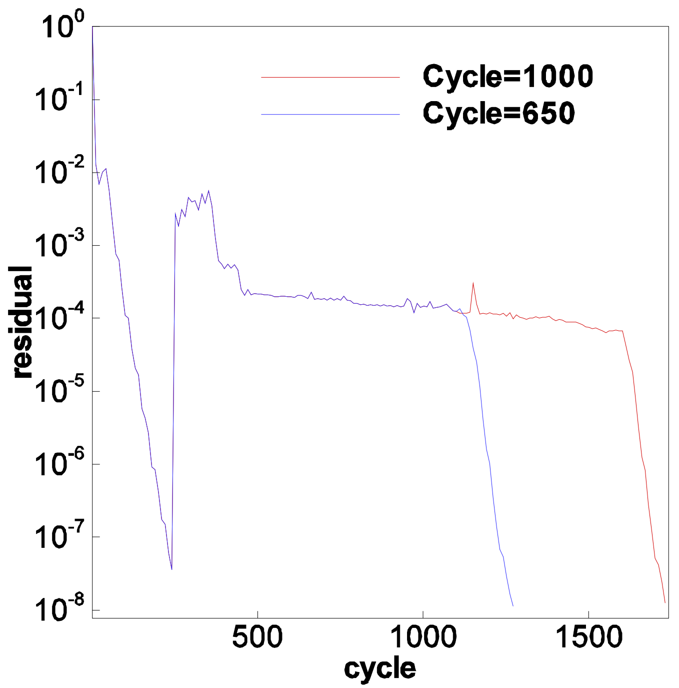


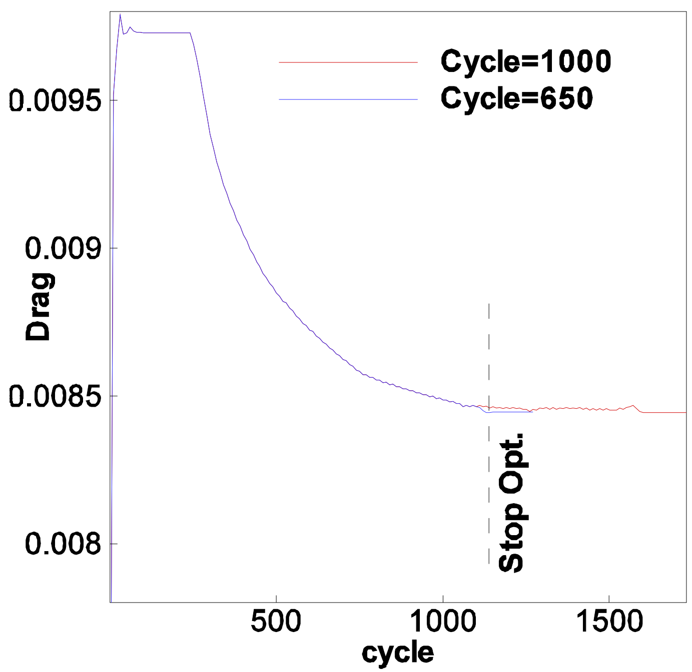
optimized

Profiles and pressure at different cross sections



Convergence history

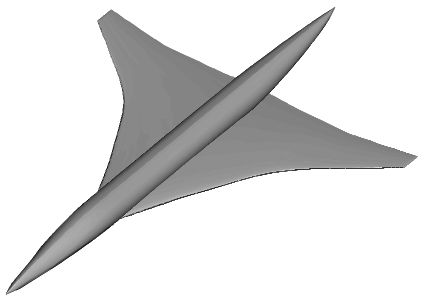




Volker Schulz

University of

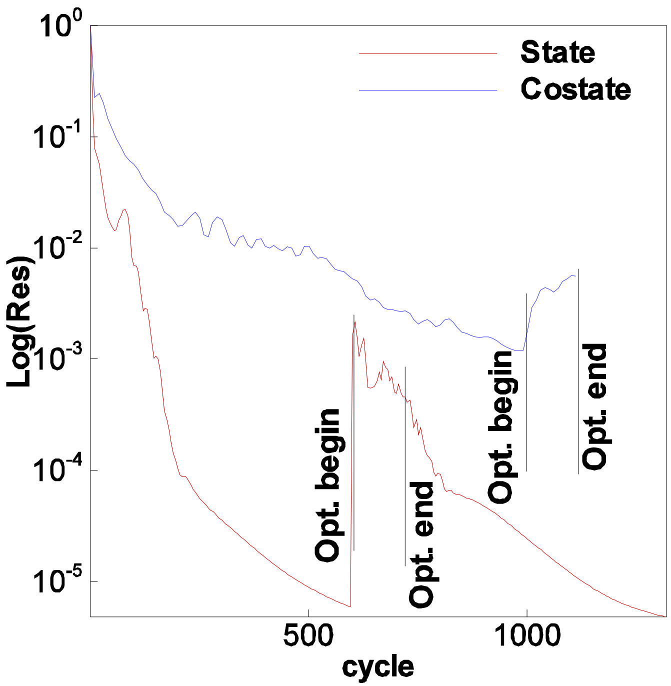
SCT body optimization

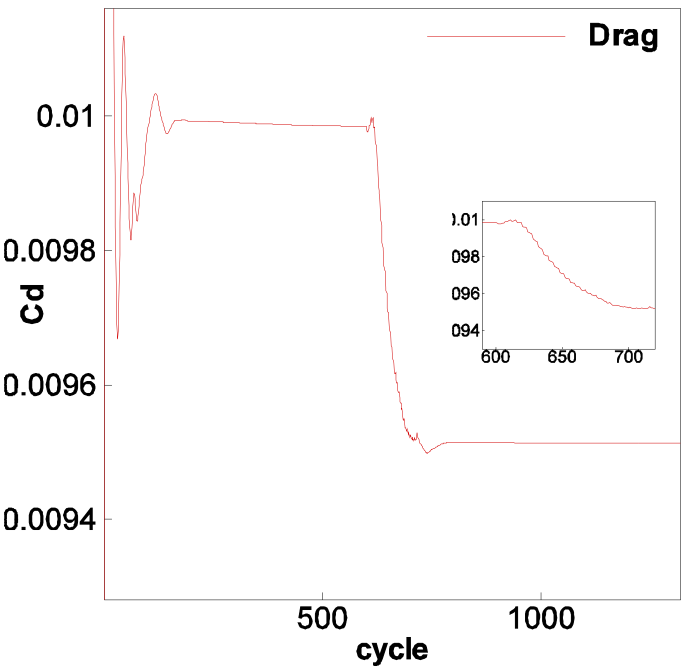


- Body only
- Minimize drag subject to constant lift
- Drag reduced by 4%
- Grid: ~ 2*10^5 grid nodes
- 10 geometry parameters: 10 radii

  additional DOF: angle of attack

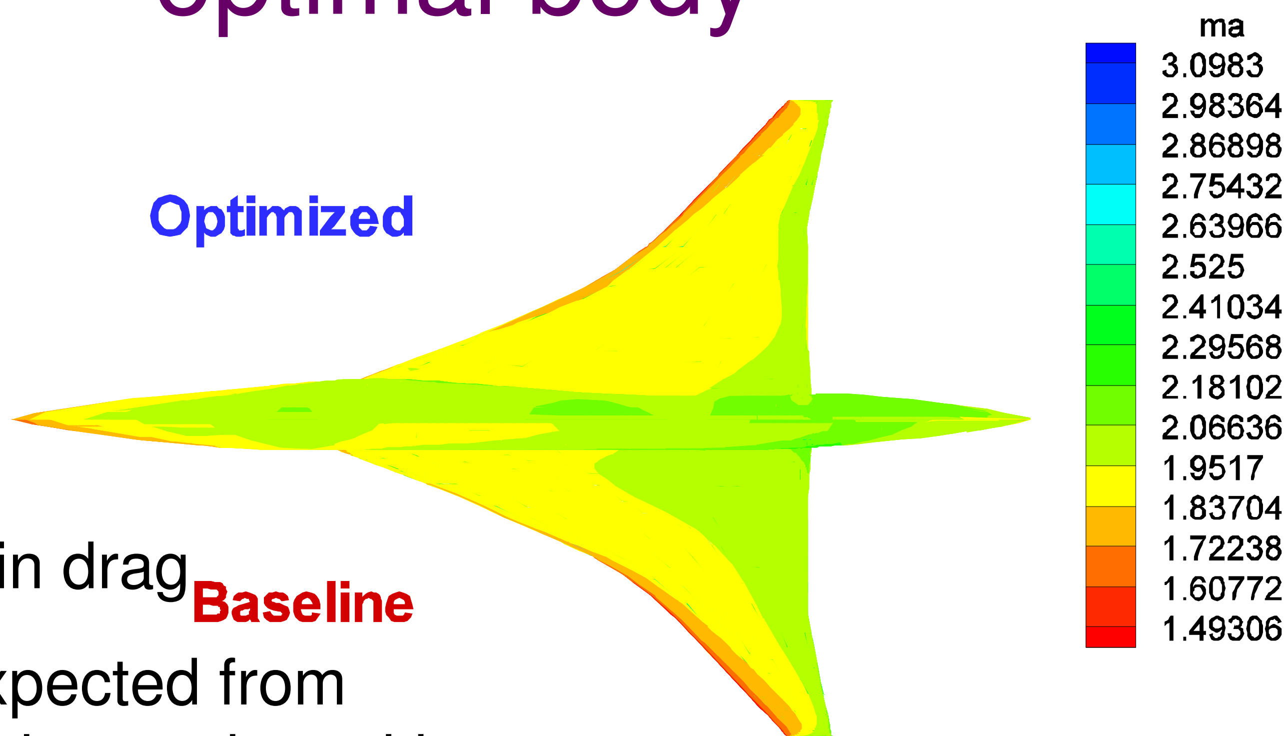
Convergence history





90 optimization cycles/ drag reduction by 4%

Combining optimal wing with optimal body



11% reduction in drag

More can be expected from optimization body together with wing

-> implementational issues…

Conclusions

- one-shot optimization based on reduced SQP ideas
- Overall computational complexity is reduced considerably.
- Limiting factor so far: frequent design space necessitate freqent calls to CAD