SAMSI summer school 2016:

PDE Optimization in HPC and Applications

Volker Schulz

Universität Trier

ALGORITHMIC OPTIMIZATION

Part1 - Overview

- The MGOPT fruitfly
- MG on stream processors
- DD for larger problems
- Results and conclusions

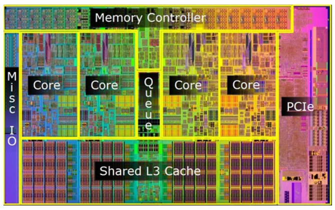
Christian Wagner: *Multigrid Optimization Methods for High Performance Computing*, PhD thesis, Trier university, 2012

S. Schmidt and V. Schulz: *A 2589 Line Topology Optimization Code Written for the Graphics Card.* Computing and Visualization in Science. Vol. 14, pages 249-256, 2011

HPC via GPGPU

- GPGPU - general-purpose computing on graphics processing unit
- GPU as testbed for multicore processors
- SIMD rather than MIMD

Fermi GPU



Quadcore CPU

Fruitfly of PDE constrained optimization

$$\begin{aligned} \min \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega)} &+ \frac{\mu}{2}\|u\|^2_{L^2(\Omega)} \\ \text{s.t.} \quad -\Delta y &= u\,, \text{ in } \Omega \\ y &= 0\,, \text{ on } \partial\Omega \end{aligned}$$

$$u \in L^2(\Omega), \qquad y \in H^1_0(\Omega),$$
$$y_d \in L^2(\Omega), \qquad \mu > 0$$

NOC

$$\boxed{\begin{array}{rcl} -\Delta y &=& u \quad \text{in } \Omega \\ y &=& 0 \quad \text{on } \partial\Omega \end{array}}$$
state

$$\begin{array}{rcl} -\Delta \lambda &=& y - y_d \quad \text{in } \Omega \\ \lambda &=& 0 \quad \text{on } \partial\Omega \end{array}$$
adjoint

$$\lambda = \mu u$$
design

Saddle point problem on a GPU

Matrix form

$$\begin{bmatrix} I & 0 & -\Delta \\ 0 & \mu I & -I \\ -\Delta & -I & 0 \end{bmatrix} \begin{pmatrix} y \\ u \\ \lambda \end{pmatrix} = \begin{pmatrix} y_d \\ 0 \\ 0 \end{pmatrix}$$

shuffled

$$\left[\begin{array}{c|cc} \mu I & -I & 0 \\ \hline -I & 0 & -\Delta \\ 0 & -\Delta & I \end{array}\right] \begin{pmatrix} u \\ \lambda \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ y_d \end{pmatrix}$$

Schur complementing gives

$$\begin{bmatrix} -\frac{1}{\mu}I & -\Delta \\ -\Delta & I \end{bmatrix} \begin{pmatrix} \lambda \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ y_d \end{pmatrix}$$

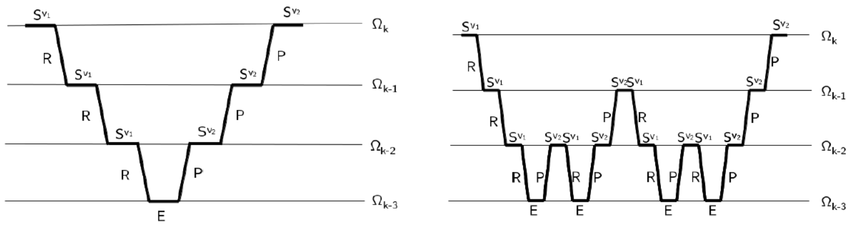
Problem: ill-conditioned for $\mu \longrightarrow 0$

Remedy: „Borzi-Trick“

$$\begin{bmatrix} -I & -\mu\Delta \\ -\Delta & I \end{bmatrix} \begin{pmatrix} \lambda \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ y_d \end{pmatrix}$$

Multigrid methods

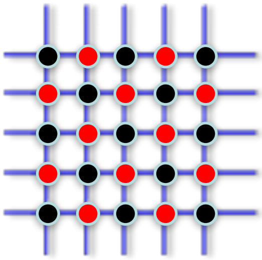
- linear algorithmic complexity
- Smoother typically related to an iterative solution scheme
- Intergrid transfer operators: prolongation/ restriction



(a) Multigrid V-cycle (b) Multigrid W-cycle

Collective Gauß-Seidel smoother

$$\begin{bmatrix} -\Delta_h & I \\ -I & -\mu\Delta_h \end{bmatrix} \begin{pmatrix} \lambda \\ y \end{pmatrix} = \begin{pmatrix} y_d \\ 0 \end{pmatrix}$$



Here red-black CGS for parallel usage on a structured grid

Obvious grid transfer operators on tensor grids

GPU implementation

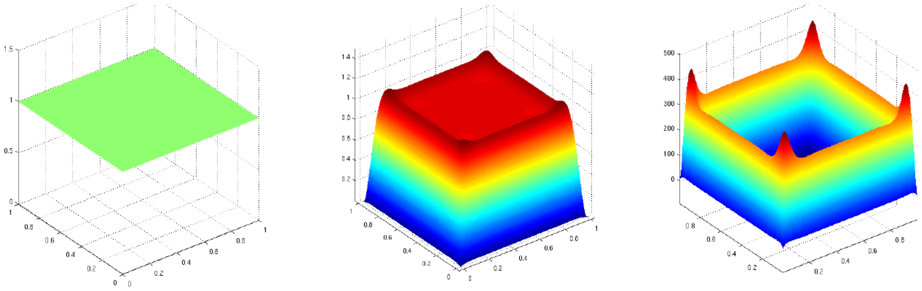
- Stream-Processor: NVIDIA-GPU GeForce GTX 295
- max. 512 threads, SIMD scheme on each core (=32 threads)

- minimize data transfer between host and device
- ensure that global memory accesses are coalesced
- use shared memory to avoid redundant transfers from global memory
- avoid different execution paths within the same warp
- decompose data in subblocks (16x16) and process them in parallel.

2D results

1024x1024 grid, $\mu = 10^{-6}$

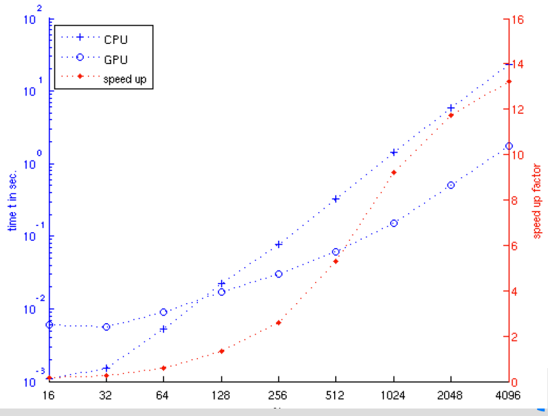
objective state control



	$\alpha = 10^{-6}$				
N	n	q	CPUt(s)	GPUt(s)	X
16	8	0.0930	0.0011	0.0060	0.18
32	7	0.0666	0.0015	0.0057	0.26
64	8	0.0868	0.0053	0.0090	0.59
128	8	0.0941	0.0225	0.0168	1.34
256	8	0.0961	0.0780	0.0304	2.57
512	8	0.0969	0.3287	0.0620	5.30
1024	9	0.0976	1.4103	0.1530	9.22
2048	8	0.0983	5.8517	0.4996	11.71
4096	8	0.0980	23.4719	1.7778	13.20

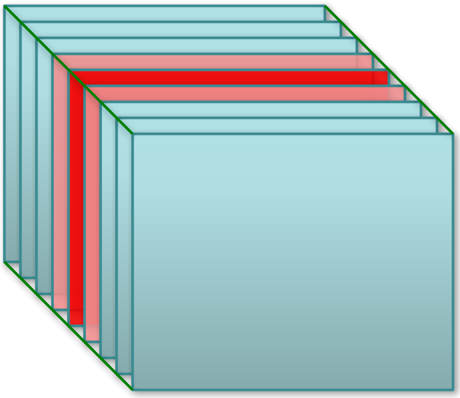
33.6 Mio unknowns

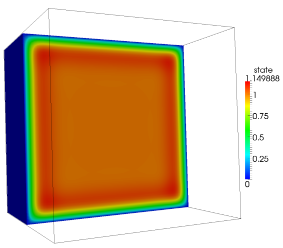
- Quadcore INTEL XEON E5620@2.40 CPU (12GB RAM)
- NVIDIA GeForce GTX 470 GPU (1280 MB RAM)

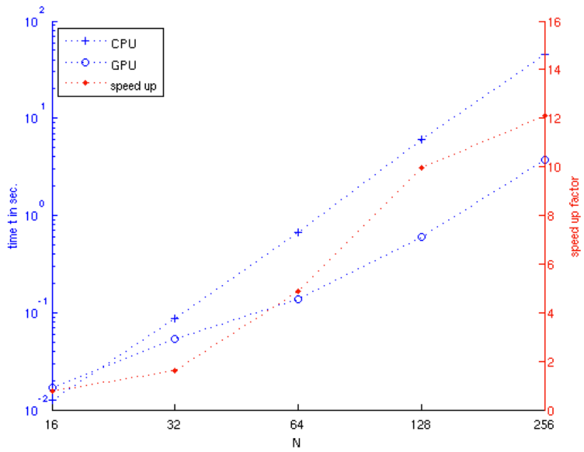


3D computations

Scrolling through the 3D grid with 3 slices







			$\alpha = 10^{-}$
N	n	q	CPUt(s)
16	13	0.2161	0.0168
32	13	0.2256	0.1046
64	13	0.2378	0.6995
128	14	0.2466	6.4941
256	14	0.2540	49.8577

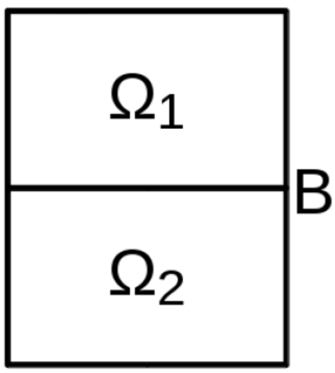
	$\alpha = 10^{-6}$					$\alpha = 10^{-8}$				
N	n	q	CPUt(s)	GPUt(s)	X	n	q	CPUt(s)	GPUt(s)	X
16	9	0.0968	0.0134	0.0149	0.90	3	0.0008	0.0051	0.0052	0.98
32	10	0.1580	0.0708	0.0442	1.60	6	0.0049	0.0443	0.0276	1.61
64	11	0.1762	0.5976	0.1239	4.82	9	0.1155	0.4937	0.1050	4.70
128	12	0.2009	5.5688	0.5592	9.96	11	0.1509	4.6276	0.4792	9.66
256	12	0.2090	42.4642	3.5228	12.05	11	0.1870	39.0655	3.2701	11.95

33.6 Mio unknowns

Size matters on GPU

- Problem: memory limitations on GPU (1280MB)
- Multiple GPU with slow interconnection
- Remedy: domain decomposition

Non-overlapping DD



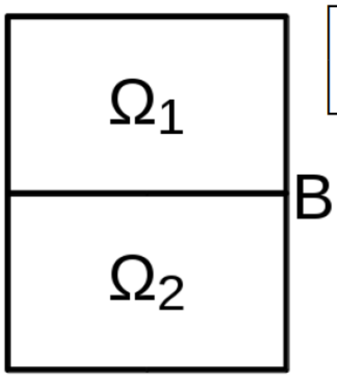
$$\begin{aligned} \mathcal{A}w &= \phi && \text{in } \Omega \\ w &= 0 && \text{auf } \partial\Omega \end{aligned}$$

$$\begin{aligned} \mathcal{A}w_1 &= \phi_1 && \text{in } \Omega_1 \\ w_1 &= 0 && \text{auf } \partial\Omega_1 \setminus B \\ w_1 &= w_B && \text{auf } B \end{aligned} \qquad \begin{aligned} \mathcal{A}w_2 &= \phi_1 && \text{in } \Omega_2 \\ w_2 &= 0 && \text{auf } \partial\Omega_2 \setminus B \\ w_2 &= w_B && \text{auf } B \end{aligned}$$

$$\frac{\partial y_1}{\partial n_1} = -\frac{\partial y_2}{\partial n_2}, \frac{\partial p_1}{\partial n_1} = -\frac{\partial p_2}{\partial n_2}$$

Steklov-Poincare operator / Dirichlet to Neumann map

DD as Schur complement



$$\begin{bmatrix} \mathcal{A}_1 & 0 & \mathcal{A}_{1B} \\ 0 & \mathcal{A}_2 & \mathcal{A}_{2B} \\ \mathcal{A}_{1B}^T & \mathcal{A}_{2B}^T & \mathcal{A}_B \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_B \end{pmatrix} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_B \end{pmatrix}$$

$$S w_B = \tilde{\phi}_B$$

Discrete S-P-operator

$$S = \mathcal{A}_B - \mathcal{A}_{1B}^T \mathcal{A}_1^{-1} \mathcal{A}_{1B} - \mathcal{A}_{2B}^T \mathcal{A}_2^{-1} \mathcal{A}_{2B}$$
$$\tilde{\phi}_B = \phi_B - \mathcal{A}_{1B}^T \mathcal{A}_1^{-1} \phi_1 - \mathcal{A}_{2B}^T \mathcal{A}_2^{-1} \phi_2$$

Aproximation of S

- S inherits structures from full space operator: p.d. but non-symmetric
- GMRES results in a slow overall algorithm
- Idea: generalize direct decomposition for poisson problem [T. F. Chan and D. C. Resasco. A domain-decomposed fast Poisson solver on a rectangle. Technical Report /DCS/RR-409, Yale University, 1985]

Theorem (Wagner 2012)

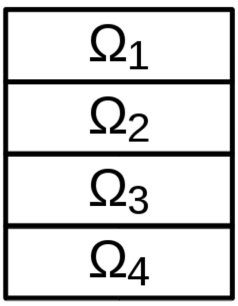
The Schur complement S can be decomposed as

$$\boxed{S = F \Lambda F}$$

$$F = \begin{bmatrix} \tilde{F} & 0 \\ 0 & \tilde{F} \end{bmatrix} \qquad \tilde{F}_{ij} = \sqrt{2h} \sin(ij\pi h)$$

Λ tridiagonal matrix from solution of 2nd order difference equation on the interface

Multiple subdomains



Schur complement of the form

$$\begin{bmatrix} H_1 & Q_2 & & \\ Q_2 & H_2 & \ddots & \\ & \ddots & \ddots & Q_{k-1} \\ & & Q_{k-1} & H_{k-1} \end{bmatrix}$$

• Simultaneous approach with complexity constant in #p wins over
• recursive approach with complexity linear in #p

Numerical results with DD

4 domains on 4 GPU versus 4 CPU cores

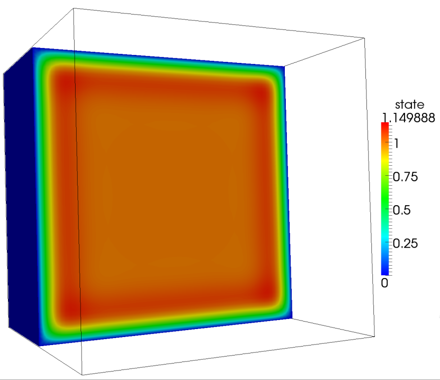
grid	CPU [s]	GPU [s]	X
1024x1024	1.72	2.74	0.63
2048x2048	6.42	4.50	1.43
4096x4096	37.47	6.68	5.61
8192x8192	143.68	14.39	10.03

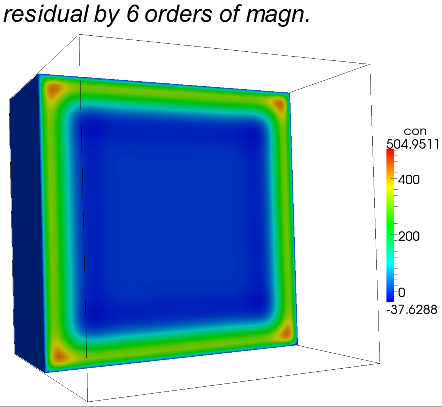
#unknowns up to 134.2 Mio

3D performance

n	# unknowns	CPU	GPU
64	0.3 Mio	00:21.897	00:01.459
128	2.1 Mio	01:52.157	00:07.477
256	16.7 Mio	14:00.305	00:56.020

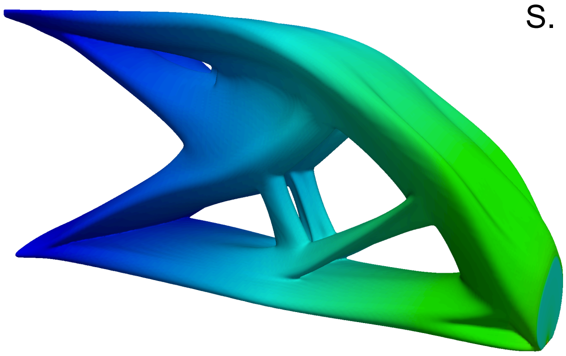
reduction of residual by 6 orders of magn.





Topology optimization on GPU

S. Schmidt, UT



180 x 180 x 360 mesh
46.5 Mio unknowns

GPU acceleration
factor 60 (core-it: CG)

Part 1 - Conclusions

- GPU implementation of MGOPT
- DD for multiple GPU
- Novel explicit Schur complement decomposition

- GPU can accelerate MGOPT by on order of magnitude

Part 2 – shape - Outline

- Motivating applications
- shape Newton methods
- shape SQP methods

S. Schmidt, C. Ilic, V. Schulz and N. Gauger: *Three dimensional large scale aerodynamic shape optimization based on the shape calculus,* AIAA Journal, Vol. 51, No. 11 (2013), pp. 2615-2627.

A. Nägel, V. Schulz, M. Siebenborn, and G. Wittum: *Scalable shape optimization methods for structured inverse modeling in 3D diffusive processes.* Computing and Visualization in Science, vol. 17, pp. 79-88, 2015

V. Schulz, M. Siebenborn and K. Welker: *Efficient PDE constrained shape optimization based on Steklov-Poincaré type metrics.* arXiv:1506.02244, 2016 (to appear in SIOPT)

Inverse skin modeling

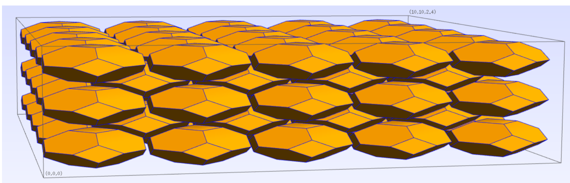
- Find shape best approximating data
- Challenge: very thin channels between cells
- Checkpointing in time

$$\min_{y,\Omega,k_1,k_1} \frac{1}{2} \int_0^T \int_\Omega \|y - \bar{y}\|^2 dxdt$$

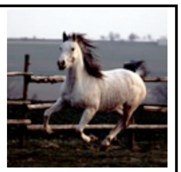
$$\text{s.t.} \quad \frac{\partial y}{\partial t} - \nabla \cdot (k \nabla y) = 0$$

$$k = \begin{cases} k_1, & \text{intracellular} \\ k_2, & \text{extracellular} \end{cases}$$

& B.C.

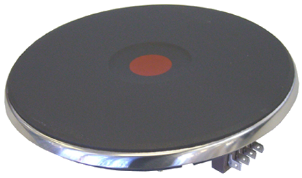


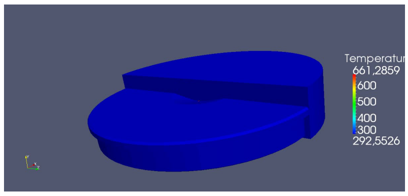
Shape Optimization of a Hot Plate

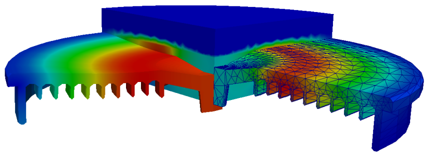


joint project with E.G.O. Elektrogerätebau GmbH, Oberderdingen

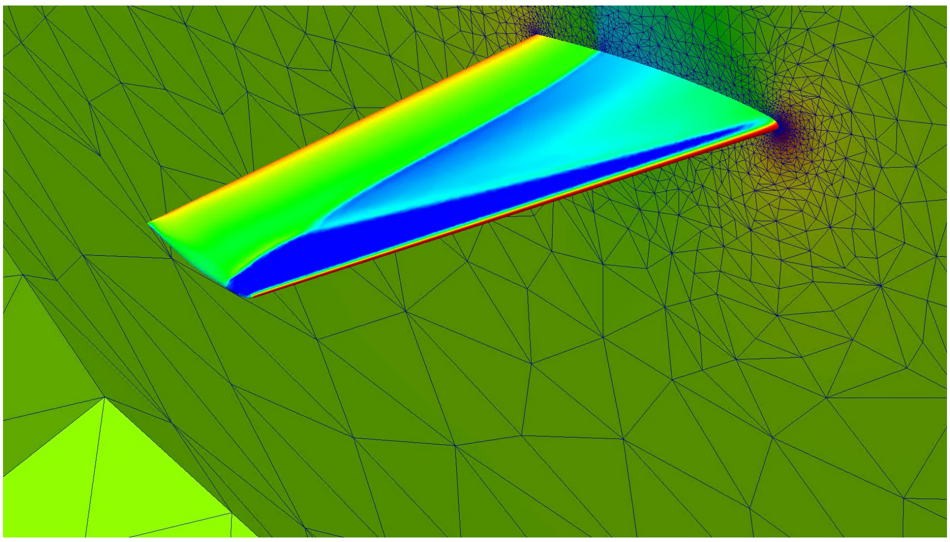
ASIL
BMBF-HPC



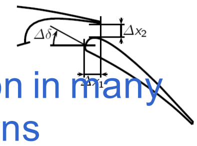
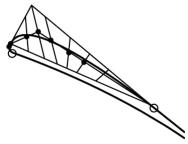




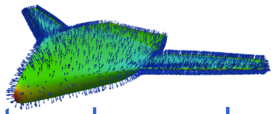
Aerodynamic shape optimization



Parametric versus nonparametric

- Finite shape parametrization in many industrial shape optimizations
  - Pro: vector space setting, fits in CAD framework
  - Con: complexity inevitably increases with number of parameters, mesh sensitivities can become expensive, set of reachable shapes is restricted



- Nonparametric approach built on shape calculus
  - Pro: avoids cons of parametric approach, can be very efficient
  - Con: no longer vector space setting, theoretically more challenging

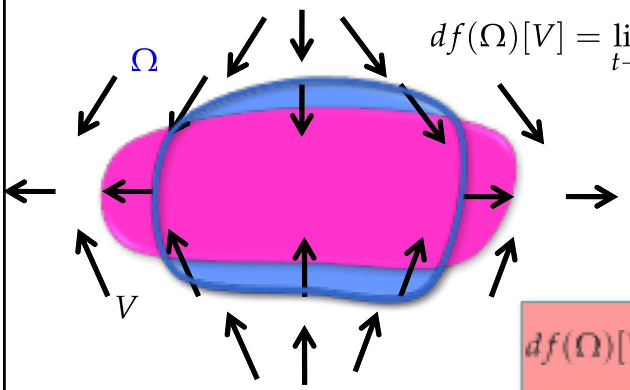
Framework

- Ongoing collaboration with German Aerospace Center in Braunschweig (MEGADESIGN, MUNA, SPP1253, ComFliTe, DGHPOPT)

- Flow solvers (Flower, Tau) are matrix-free and based on multigrid methods featuring polynomial smoothers (aka RK)

  [for Euler-eq. cf. van Leer/Tai/Powell AIAA CFD 1989]

Shape gradients for free node parametrizations



$$df(\Omega)[V] = \lim_{t \to 0} \frac{f(T_{t,V}(\Omega)) - f(\Omega)}{t}$$

$$df(\Omega)[V] = (\nabla f, < V, n >)_{\partial\Omega}$$

$T_{t,V}(\Omega)$, where $T_{t,V}(x) = x + t \cdot V(x)$

(Hadamard)

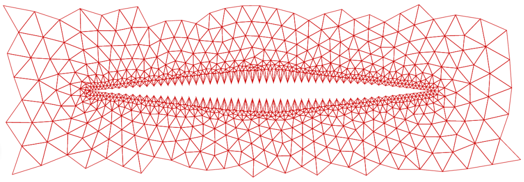
Further names to be mentioned in the field of shape calculus:

Zolesio, Haslinger, Sokolowski, Pironneau, Mohammadi, Delfour, Neittanmäki, Berggren, Hintermüller, Ring, Eppler, Harbrecht, Zuazua,…

Potential problem: too much freedom

- Usage of shape derivatives alone may lead to unphysical geometries

$$H\Delta p = -\nabla \hat{f}(p^k)$$



- Shape Hessian approximations help to
  - „smooth“ gradients
  - Speed up convergence in the fashion of a Newton-like method
  - Give potentially mesh independence

Example: drag in volume formulation (dissipation of kinetic energy into heat)

Objective

$$\min_{(u,p,\Omega)} \dot{E}(u,\Omega) := \frac{1}{2}\int_\Omega \nu \sum_{j,k=1}^{3} \left(\frac{\partial u_k}{\partial x_j}\right)^2 dA$$

Constraints

a) Stokes Equation

$$-\nu\Delta u + \nabla p = 0$$
$$\text{div } u = 0$$

b) Navier-Stokes Equation

$$-\nu\Delta u + \rho u \nabla u + \nabla p = 0$$
$$\text{div } u = 0$$

Shape derivatives

(Schmidt/S.: Control and Cybernetics, 2010)

a) Stokes

$$d\dot{E}_S(u,\Omega)[V] = \int_\Gamma \langle V,n\rangle \left[-\nu \sum_{k=1}^{3}\left(\frac{\partial u_k}{\partial n}\right)^2\right] dS$$

b) Navier-Stokes

$$d\dot{E}_{NS}(u,\Omega)[V] = \int_\Gamma \langle V,n\rangle \left[\nu \sum_{k=1}^{3}\left(\frac{\partial u_k}{\partial n}\right)^2 - \frac{\partial u_k}{\partial n}\frac{\partial \lambda_k}{\partial n}\right] dS$$

$$\begin{aligned} -\nu\Delta\lambda + \rho\lambda\nabla u - \rho\left(\nabla\lambda\right)^T u + \nabla\lambda_p &= -2\Delta u \quad \text{in } \Omega \\ \text{div } \lambda_p &= 0 \quad \text{in } \Omega \end{aligned}$$

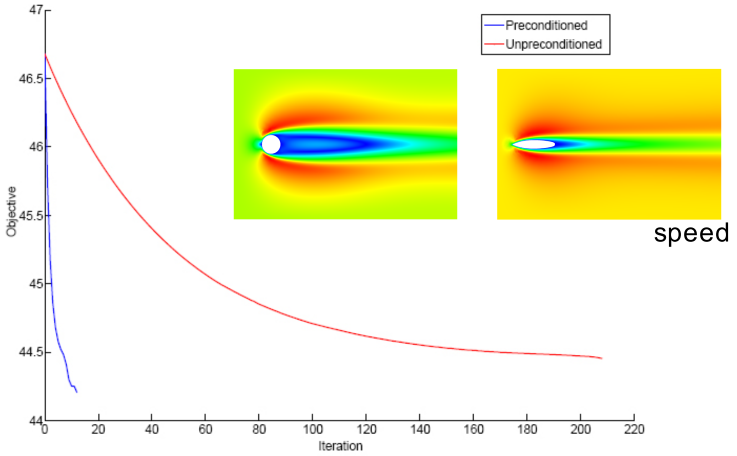
Shape Hessians

$$d^2J(u,\Omega)[V_1,V_2]$$

- Can become rather cumbersome and the numerical interpretation of the operators is difficult (Eppler/S./Schmidt, JOTA 2008)
- Fourier mode analysis gives an idea of the structure of a good approximation(S./Schmidt, SICON 2009)

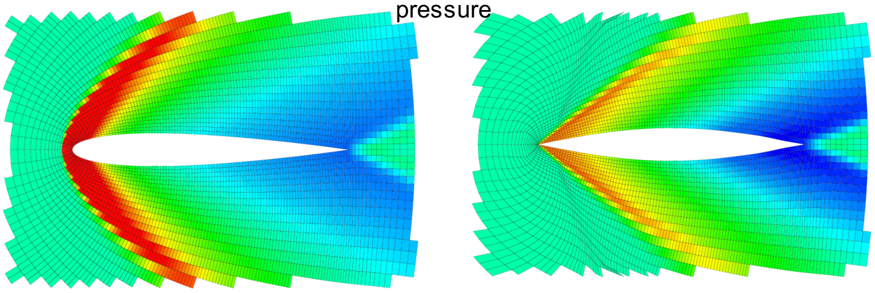
$$\widetilde{H} = -\alpha\Delta_\Gamma + id \quad \text{with symbol } \Sigma = 1 + \alpha\omega^2$$

Performance for Navier-Stokes



12 vs. 200: 96% less iterations

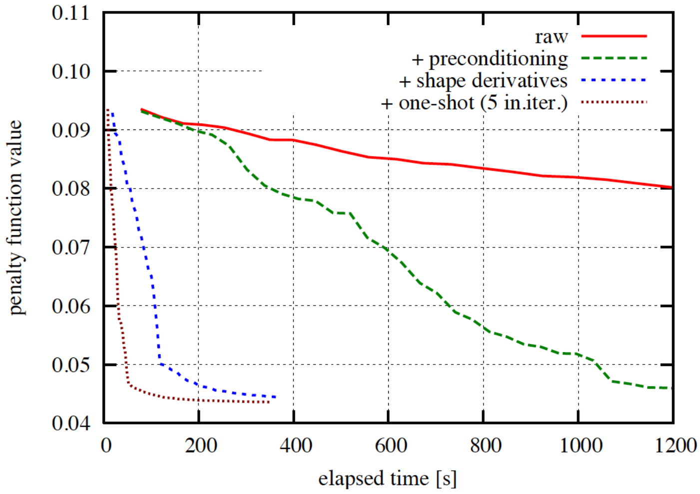
Optimal non-parametric design for Euler flow in TAU (DLR)



From NACA0012 to Haack Ogive

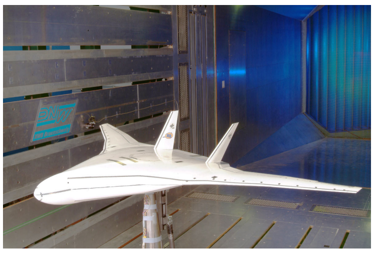
Mach 2.0 strong detached bow shock transformed to weak
Drag reduction 45%

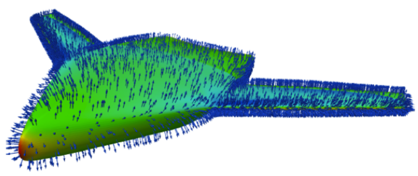
Overall performance



Wall clock time reduced by 99% (2.77h versus 100s)

Vela aircraft (joint with DLR)

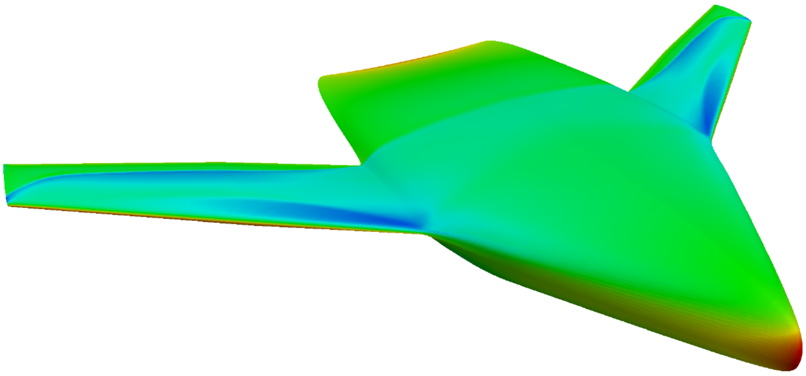




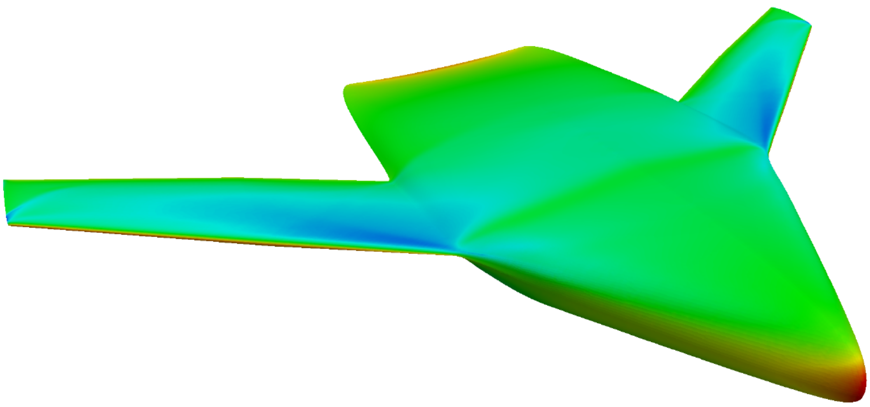
- VELA: Very efficient large aircraft
- Design study for blended wing-body configurations

- 115,673 surface nodes to be optimized
- Planform constant

[Schmidt/Ilic/Gauger/S. AIAA Journal 2013]



Shape	State	C_D	C_L	α	M_∞
115,673	29,297,175	$4.770 \cdot 10^{-3}$	$1.787 \cdot 10^{-1}$	1.8°	0.85



Shape	C_D	%	C_L	%
115,673	$3.342 \cdot 10^{-3}$	-30.06%	$1.775 \cdot 10^{-1}$	-0.67%

A closer look at the shape Hessian

- Symmetry $\boxed{\text{No}}$

$$d(df(\Omega)[W])[V] = \int_{\partial\Omega} \left(\frac{\partial g}{\partial \vec{n}} + \kappa_c g\right) \langle W, \vec{n}\rangle \langle V, \vec{n}\rangle + g \langle DW\,V, \vec{n}\rangle \, ds$$

- Taylor series

$$f(p) = f(p_o) + df(p_o)(p - p_o) + \frac{1}{2} d^2 f(p_o)(p - p_o, p - p_o) + \mathcal{O}(\|p - p_o\|^3)$$

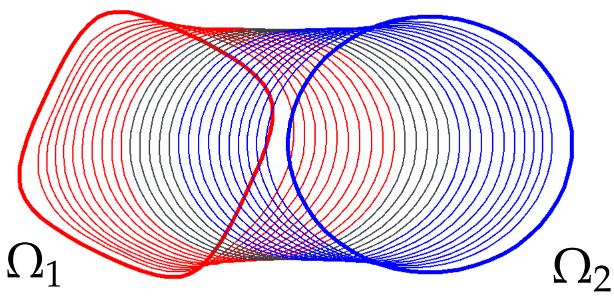
?

$\Rightarrow$ Sufficient conditions

$\Rightarrow$ Quadratic convergence of Newton method

- Distance concepts

„Morphing“: Riemannian length of shortest connecting path



The Riemannian metric of Michor and Mumford (2006)

Shape set $B_e(S^1, \mathbb{R}^2) = \mathrm{Emb}(S^1, \mathbb{R}^2)/\mathrm{Diff}(S^1)$

is a manifold with tangent space

$$T_c B_e \cong \{h \mid h = \alpha \vec{n},\, \alpha \in C^\infty(S^1, \mathbb{R})\}$$

Scalar product

$$S^A(h,k) = \int_c \alpha\beta + A\alpha'\beta' ds$$

$$G^A(h,k) = \int_c (1 + A\kappa_c^2)\alpha\beta ds\,, \quad A > 0$$

defines a Riemannian manifold

A Riemannian view on shape optimization

- Defining the action of a vector field as the shape derivative, we can unleash the Riemannian structure on shape optimization

$$h(f)(c) = df(\Omega)[V]\,, \quad V = h\vec{n}\,,\, c = \partial\Omega$$

- Optimization on manifolds can be performed as in [Absil 2008] for matrix manifolds.

Consequences

- Covariant derivative (here for curvature metric)

$$\nabla_V W = \langle DW\,V, \vec{n}\rangle + \frac{1}{2}(\kappa_c + \frac{2A\kappa_c^3}{1+A\kappa_c^2})\,\langle V, \vec{n}\rangle\,\langle W, \vec{n}\rangle$$
$$+ A\kappa_c\,(\langle V, \vec{n}\rangle\,\langle W, \vec{n}\rangle)_{\tau\tau}$$

- Symmetric Riemannian shape Hessian

$$G^A(\mathrm{Hess} f(\Omega)[V], W) := G^A(\nabla_V \mathrm{grad} f(\Omega), W)$$
$$= d(df(\Omega)[W])[V] - df(\Omega)[\nabla_V W]$$

- Taylor series expansion

$$f(\exp_\Omega(h)) = f(\Omega) + G^A(\mathrm{grad} f(\Omega), h) + \frac{1}{2}(\mathrm{Hess} f(\Omega)h, h) + \mathcal{O}(\|h\|^3)$$

Newton convergence

- Riemannian variants of quadratic convergence results are possible.
- In particular, if we eliminate expressions from the Hessian, which are zero at the solution, also quadratic convergence is possible.

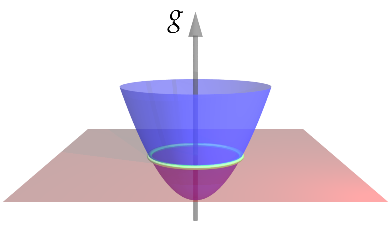
Optimization algorithms

Retraction
$\partial\Omega^{k+1} = \partial\Omega^k + \alpha d^k$

$$\Omega^{k+1} = \exp_{\Omega^k}(\alpha d^k)$$
$$d^k = -\mathrm{grad} f(\Omega^k) \text{ or } d^k = -\mathrm{Hess}(\Omega^k)^{-1}\mathrm{grad} f(\Omega^k)$$

Steepest descent	Newton method
linear conv.	quadratic conv.

Back to the simple example



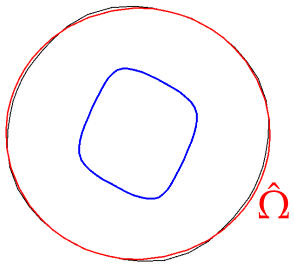
$$\min_{\Omega} f(\Omega) := \int_{\Omega} g(x)dx$$
$$\text{with } g(x) = x^\top x - 1$$

2

Riemannian shape Hessian

$$\begin{aligned} G^A(\mathrm{Hess} f(\Omega)[V], W) &= d(df(\Omega)[W])[V] - df(\Omega)[\nabla_V W] \\ &= \int_{\partial\Omega} \left(\frac{\partial g}{\partial \vec{n}} + \frac{\kappa_c}{2} g - \frac{gA\kappa_c^3}{1 + A\kappa_c^2}\right)) \langle V, \vec{n}\rangle \langle W, \vec{n}\rangle \, ds \\ &\quad - \int_{\partial\Omega} gA\kappa_c \left(\langle V, \vec{n}\rangle \langle W, \vec{n}\rangle\right)_{\tau\tau} ds \end{aligned}$$

Performance of optimization algorithms



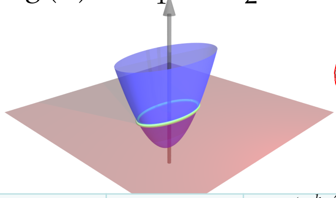
Steepest descent with exact linesearch

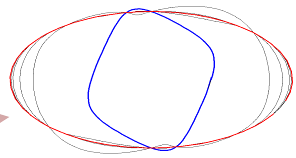
Iteration	objective	$d^A(\Omega^k,\hat{\Omega})$	line search
0	-2.66624	3.4135	0.69
1	-4.18749	7.2458 E-02	0.50
2	-4.18879	6.4371 E-04	0.50
3	-4.18879	7.0061 E-08	

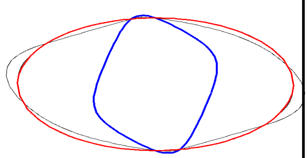
Equivalent to Newton method

-> example too simple

$$g(x) = x_1^2 + 4x_2^2$$







Iteration	SD-objective	$d_{SC}(\Omega^k,\hat{\Omega})$	SD-line search	NM-objective	$d_{NM}(\Omega^k,\hat{\Omega})$	NM-line search
0	-0.5571	**.9222E+00**	.5100E+00	-0.5571	**.9222E+00**	0.6400
1	-0.7630	**.2142E+00**	.3100E+00	-0.7775	**.1359E+00**	1.0000
2	-0.7830	**.5882E-01**	.3700E+00	-0.7854	**.4265E-02**	1.0000
3	-0.7852	**.1662E-01**	.3200E+00	-0.7854	**.1037E-04**	1.0000
4	-0.7854	**.4459E-02**	.3500E+00	-0.7854	**.2590E-09**	
5	-0.7854	**.1356E-02**	.3300E+00			
6	-0.7854	**.3951E-03**	.3400E+00			
7	-0.7854	**.1218E-03**	.3300E+00			
8	-0.7854	**.3643E-04**	.3500E+00			
9	-0.7854	**.1194E-04**	.3200E+00			
10	-0.7854	**.3421E-05**	.3500E+00			
11	-0.7854	**.1137E-05**	.3200E+00			
12	-0.7854	**.3305E-06**	.3500E+00			
13	-0.7854	**.1109E-06**	.3000E+00			
14	-0.7854	**.2894E-07**				

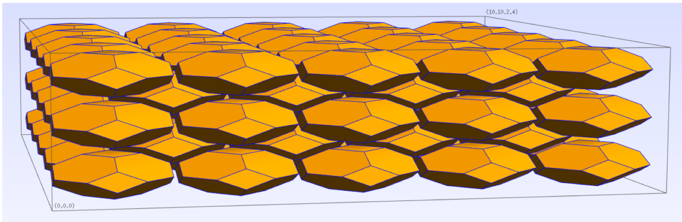
Advantages of the Riemannian view

- Symmetric shape Hessian variant
- Sufficient optimality conditions
  - Example: $\mathrm{Hess} f(\hat{\Omega}) = 2 \cdot id$ is coercive
- Analysis of convergence order of shape optimization methods
  - Quadratic convergence for Newton method can be observed

Related approaches

- Hintermüller/Ring 2003/2004: special choice of perturbations
- Hintermüller 2005: convergence from descent property
- Ring/Wirth 2012: convergence theory on Riemannian manifolds
- Sundamoorthi/Mennucci/Soatto/Yezzi 2011: Sobolev-type metric on shape space
- Harbrecht/Eppler: well-posedness studies for star shaped domains

Parabolic interface problem



- Motivation: structured inverse modeling of human skin cells
- combined estimation of coefficient and cell shapes
- discrete measurements in time and space
- millions of unknowns, especially on surfaces
- highly parallel solver based on UG4 (joined work with group of G. Wittum, GCSC Frankfurt)

$$\min_{(y,k_1,k_2,\Omega)} J(y,\Omega) = \frac{1}{2}\int_0^T\int_\Omega (y-z)^2\, d\Omega\, dt \quad + \quad \mu \int_{\Gamma_{\mathrm{int}}} 1\, dS$$

$$\text{s.t.} \quad \frac{\partial y}{\partial t} - \nabla \cdot k\nabla y = f \quad \text{in} \quad \Omega \times (0,T]$$

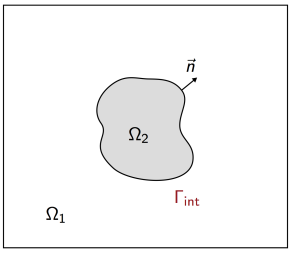
$$[\![y]\!] = 0 \quad \text{on} \quad \Gamma_{\mathrm{int}} \times (0,T]$$

$$\left[\!\!\left[k\frac{\partial y}{\partial \vec{n}}\right]\!\!\right] = 0 \quad \text{on} \quad \Gamma_{\mathrm{int}} \times (0,T]$$

$$y = 1 \quad \text{on} \quad \Gamma_{\mathrm{top}} \times (0,T]$$

$$\frac{\partial y}{\partial \vec{n}} = 0 \quad \text{on} \quad \Gamma_{\mathrm{bottom}} \cup \Gamma_{\mathrm{left}} \cup \Gamma_{\mathrm{right}} \times (0,T]$$

$$y = y_0 \quad \text{in} \quad \Omega \times \{0\}$$



$$k = \begin{cases} k_1 & \text{in} \quad \Omega_1 \\ k_2 & \text{in} \quad \Omega_2 \end{cases}$$

Shape derivative here

- Hadamard form

$$dJ(y,\Omega)[V] = -\int_0^T \int_{\Gamma_{\text{int}}} [\![k]\!] \langle \nabla y_2, \nabla p_1 \rangle \langle V, \vec{n} \rangle \; dS \, dt + \int_{\Gamma_{\text{int}}} \mu\kappa \langle V, \vec{n} \rangle \, dS$$

- where *p* solves the adjoint problem

$$-\frac{\partial p}{\partial t} - \nabla \cdot k \nabla p = z - y \qquad \& \text{ b.c.}$$

- Very similar to results by Ito/Kunisch 2008 and Paganini 2014 for the elliptic case
- Only finitely many measurements z in time => no storage problems to be handled by strategies like checkpointing.

Basic approach

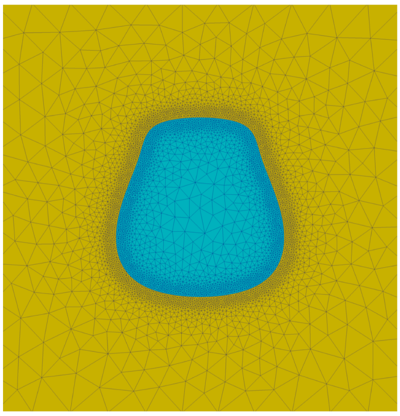
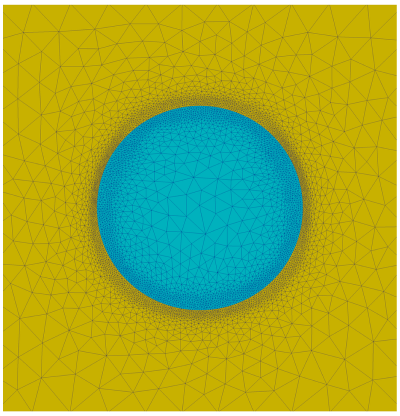
- Use shape derivative in a steepest descent manner

- In literature, mostly Riesz representations based on L^2 or Laplace-Beltrami are used, i.e.

$$d = (I - \sigma \triangle_\Gamma)^{-1} g$$

- Shape manifold framework allows even for quasi-Newton algorithms (S. JFCM, 2014)

Adaptive meshes

- shape gradient as boundary condition in linear elasticity mesh deformation
- shape calculus depends on fine grid near variable boundary

Riemannian Quasi-Newton

quasi–Newton method – k. iteration:

1. compute update formula $\mathrm{H_k}$ for $\mathrm{Hess\,J(c_k)}$ and the increment $\Delta \mathrm{c} = \mathrm{H_k^{-1} grad\,J(c_k)}$
2. increment $\mathrm{c_{k+1}} := \exp_{\mathrm{c_k}}(\alpha_{\mathrm{k}} \Delta \mathrm{c})$ for some steplength α^{k}

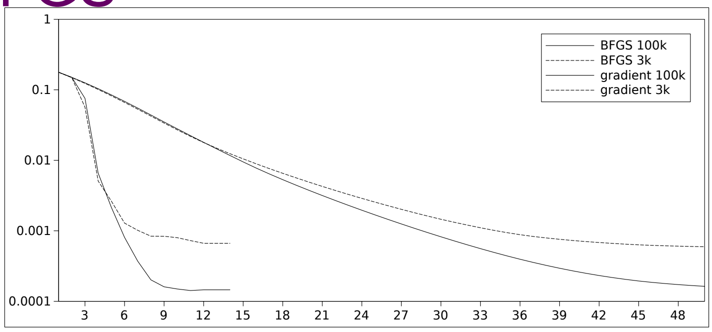
- $\mathrm{grad\,J}$ Riemannian shape gradient
- update formula $\mathrm{H_k}$ for the Riemannian shape Hessian $\mathrm{Hess\,J(c_k)}$ is based on the secant condition

$$\mathrm{H_k s_k = w_k}$$

where $\mathrm{s_k}$ denotes the difference between iterated shapes and $\mathrm{w_k}$ the difference between iterated shape gradients

Volker Schulz SAMSI Summer School 2016 Universität Trier

L-BFGS



S./Siebenborn/Welker, 2014, arXiv:1409.3464, SICON (to appear)

Noisy data do not affect method!

Scalability issues

- Hadamard form of gradient lives on the boundary only
- Laplace-Beltrami metric lives on the boundary only

- => load balancing has to take into account explicitely boundary nodes

Domain integral formulation

$$dJ(y,\Omega)[V] = \int_0^T \int_\Omega -k\nabla y^T \left(\nabla V + \nabla V^T\right) \nabla p - p\nabla f^T V$$
$$+ \operatorname{div}(V) \left(\frac{1}{2}(y-\overline{y})^2 + \dot{y}p + k\nabla y^T \nabla p - fp\right) dx\, dt.$$

- often intermediate step in derivation
- is shown as asymptotically more accurate (Berggren 2010, Paganini, 2014)
- Volumetric operations do not change load balancing in parallelization.
- => improves scalability of optimization algorithm, if scalar product can also be transferred to the volume.

Harmonizing shape metric

$$S^p : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma), \quad a(V,W) = \int_\Gamma \alpha \cdot (\gamma_0 W)^\top n ds, \ \forall\, W \in H_0^1(\Omega, \mathbb{R}^d)$$
$$\alpha \mapsto (\gamma_0 V)^\top n$$

$$\boxed{g^S(\alpha,\beta) := \langle \alpha, (S^p)^{-1}\beta \rangle = \int_\Gamma \alpha(s) \cdot [(S^p)^{-1}\beta](s) ds}$$

- S^p : projected Poincaré-Steklov
- inherits positivity from full P.-S.
- enables combination of gradient representation and mesh deformation

[S./ Siebenborn/Welker, 2015, arXiv:1506.02244]

Resulting method

- Compute mesh deformation U from elasticity equation

$$a(U,V) = DJ_\Gamma[V] = DJ_\Omega[V]\,,\ \forall\, V \in H_0^1(\Omega,\mathbb{R}^d)$$

- Use U also in l-BFGS

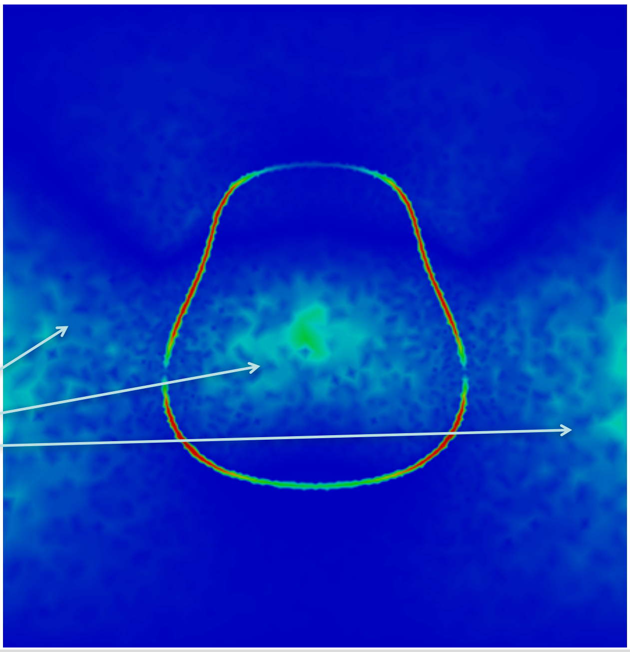
- U is typically used as boundary deformation in steepest descent

Zoom

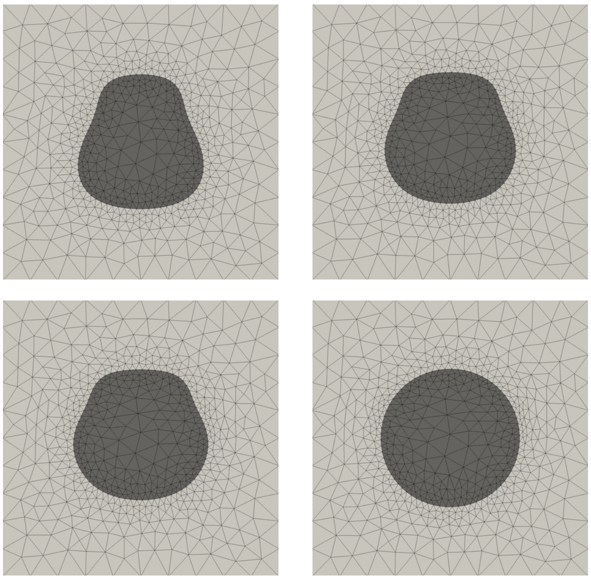
$DJ_\Omega[V_h]$

$V_h \in H_0^1(\Omega,\mathbb{R}^d)$

Force 0 outside boundary!

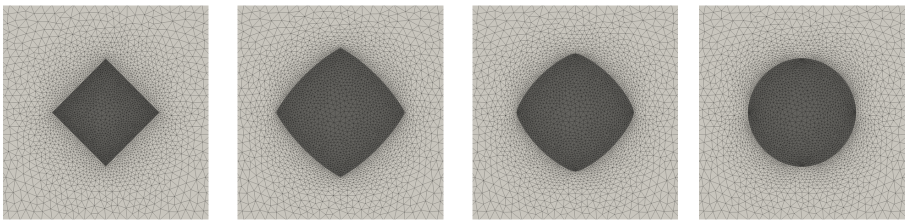


-> nice convergence even for coarse grids

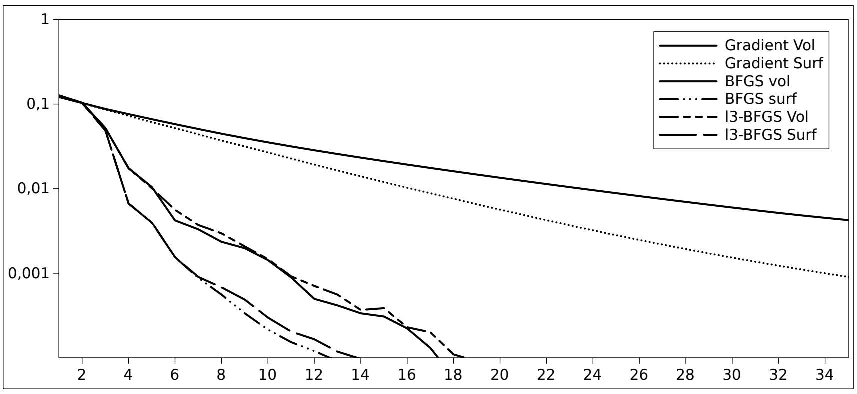


just
1000
cells

even kinky starts are possible



Convergence comparison on same grid



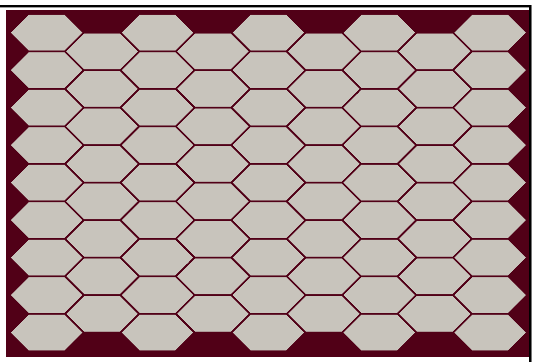
Approximative geodesic distance to solution

Algorithmic aspects

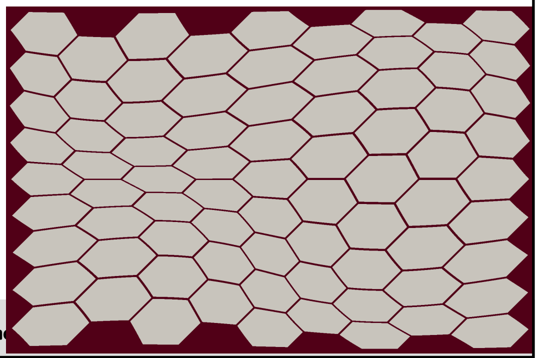
- Multigrid (ug4) is used for
  - each timestep of forward problem
  - each timestep of adjoint problem
  - evaluation of shape gradient based on Laplace-Beltrami or Steklov-Poincare metric
  - mesh deformation based on elasticity equation
- Classical forward-backward loop in each optimization iteration

Skin Problem in 2D

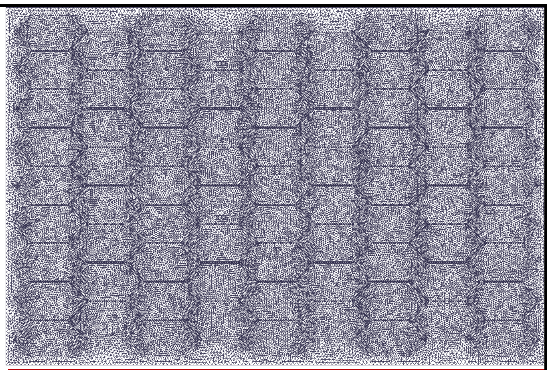
Start

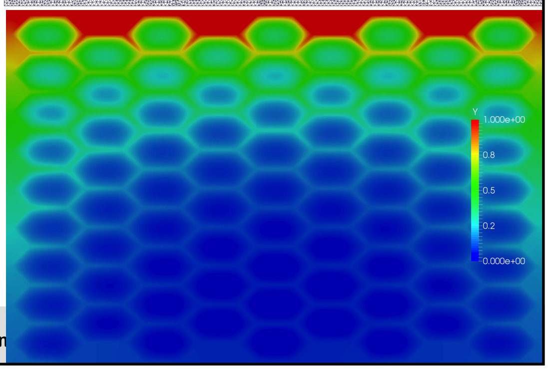


Goal

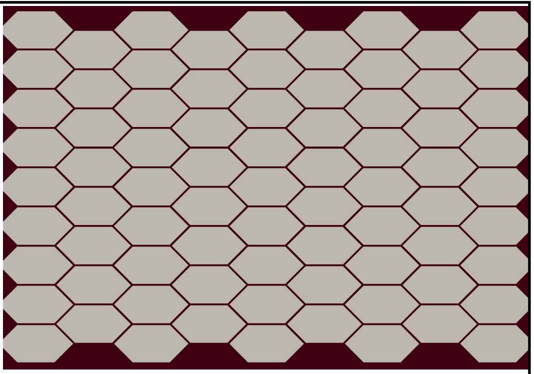


State solution

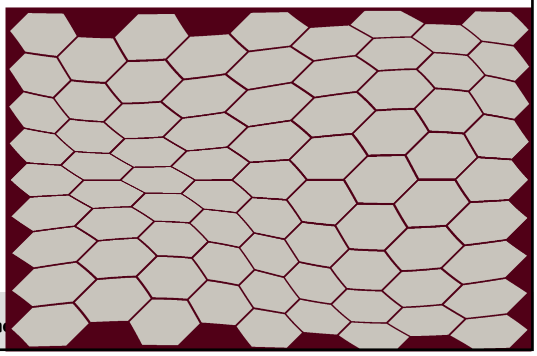




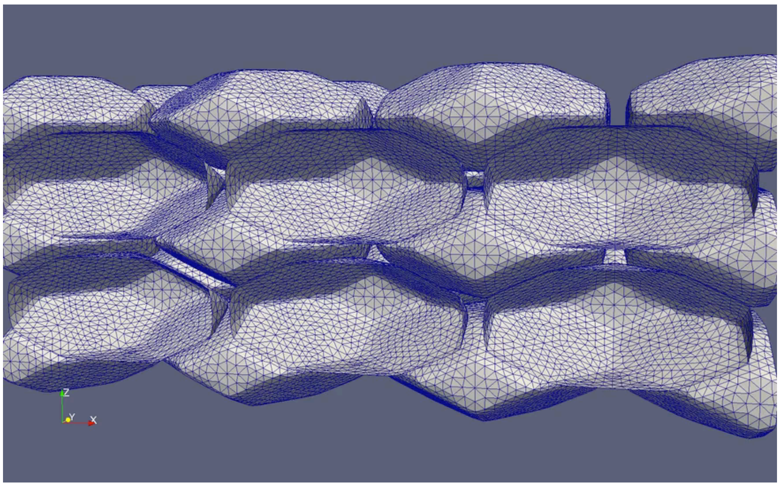
Iterations



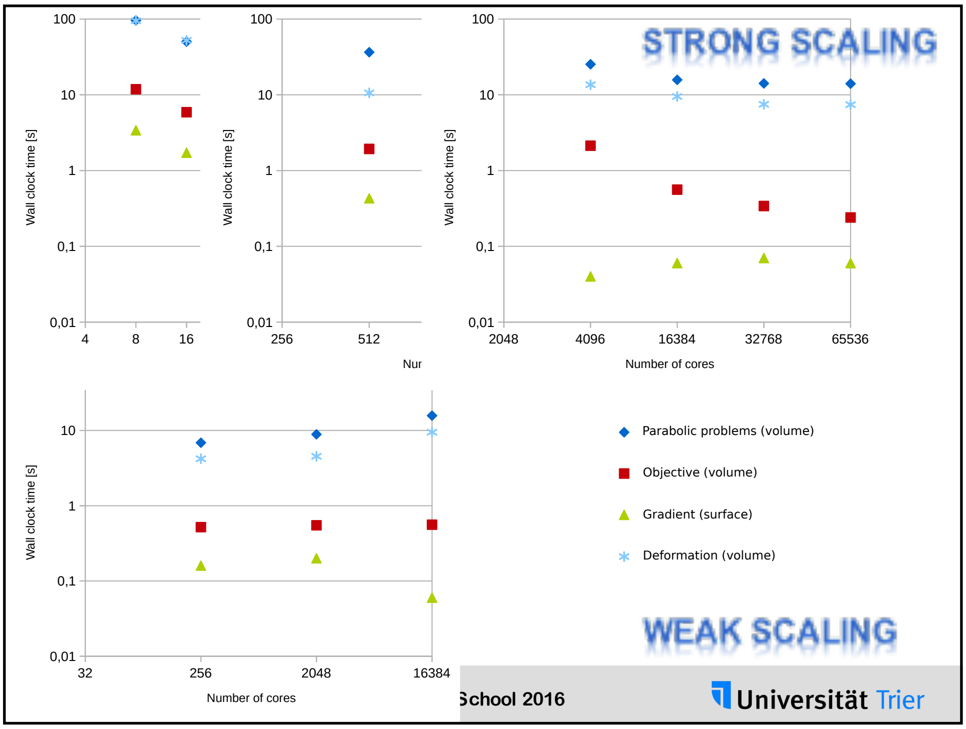
Goal



Coarse 3D-results



1.8 Mio elements, 10 time steps, 20 optimization steps



Part 2 - Conclusions

- Many applications profit from a non-parametric approach.
- The Riemannian view gives novel analytic means and enables the integration of optimal control technology into shape optimization.
- Poincaré-Steklov type metric allows less smootheness assumptions and improves scalability