Shape optimisation with the Discrete Adjoint Technique

René Schneider[1] Peter Jimack[2] Andreas Günnel[3]

[1]Institut für Numerische Mathematik
Fakultät Mathematik und Naturwissenschaften
TU Dresden

[2]School of Computing
University of Leeds

[3]Numerische Mathematik
Fakultät für Mathematik
TU Chemnitz

14 April 2010

NM

TECHNISCHE UNIVERSITÄT DRESDEN

Motivation

- In many cases a PDE system is most easily inlfuenced by the shape of its domain, e.g. flow control in tap. $\Rightarrow$ shape optimisation
- We consider only parameterised shape (finite dimensional), because this is natural for manufacturing tools.

Motivation

http://commons.wikimedia.org/wiki/File:Tap.png

$\Rightarrow$ shape optimisation

- We consider only parameterised shape (finite dimensional), because this is natural for manufacturing tools.

Motivation

- In many cases a PDE system is most easily inlfuenced by the shape of its domain, e.g. flow control in tap. $\Rightarrow$ shape optimisation
- We consider only parameterised shape (finite dimensional), because this is natural for manufacturing tools.

Motivation

- In many cases a PDE system is most easily inlfuenced by the shape of its domain, e.g. flow control in tap. $\Rightarrow$ shape optimisation
- We consider only parameterised shape (finite dimensional), because this is natural for manufacturing tools.

Shape optimisation

$$\min_{\mathcal{F}\in\mathcal{D}} \; I(u,\mathcal{F})$$

$$\begin{aligned} \text{subject to} \quad & \mathcal{L}(u) = f \quad && \text{in} \quad \Omega(\mathcal{F}) \\ \text{and} \quad & \mathcal{B}(u) = g \quad && \text{on} \quad \partial\Omega(\mathcal{F}), \end{aligned}$$

$\mathcal{F}$	the shape,
$\mathcal{D}$	set of admissible shapes,
$\mathcal{L}$	differential operator of the PDE,
$\Omega(\mathcal{F})$	PDE domain as function of the shape,
$\mathcal{B}$	operator defining boundary conditions,
f and g	given functions.

Existence of solutions, e.g. [Haslinger/Mäkinen 2003].

Sensitivity Analysis

- Consider scalar quantity

$$I(s) = \widetilde{I}(u(s), s), \quad \text{where vector } u(s) \text{ defined by}$$
$$0 = R(u(s), s). \tag{1}$$

$\Rightarrow$ require ∇I

- Consider small perturbation δs in s,

$$\delta I = \frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial s}\delta s$$

$$0 = \delta R = \frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial s}\delta s.$$

$\Rightarrow$ sensitivity equation

Sensitivity Analysis

- Consider scalar quantity

$$I(s) = \widetilde{I}(u(s), s), \quad \text{where vector } u(s) \text{ defined by}$$
$$0 = R(u(s), s). \tag{1}$$

  $\Rightarrow$ require ∇I

- Consider small perturbation δs in s,

$$\delta I = \frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial s}\delta s$$

$$0 = \delta R = \frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial s}\delta s.$$

  $\Rightarrow$ sensitivity equation

- Consider scalar quantity

$$I(s) = \widetilde{I}(u(s), s), \quad \text{where vector } u(s) \text{ defined by}$$
$$0 = R(u(s), s). \tag{1}$$

  $\Rightarrow$ require ∇I

- Consider small perturbation δs in s,

$$\delta I = \frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial s}\delta s$$

$$0 = \delta R = \frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial s}\delta s.$$

  $\Rightarrow$ sensitivity equation

Discrete Adjoint Technique

$$\delta I = \left(\frac{\partial \widetilde{I}}{\partial u} \delta u + \frac{\partial \widetilde{I}}{\partial s} \delta s \right) - \Psi^T \left(\frac{\partial R}{\partial u} \delta u + \frac{\partial R}{\partial s} \delta s \right)$$

Discrete Adjoint Technique

$$\begin{aligned}\delta I &= \left(\frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial s}\delta s\right) && - \Psi^T \left(\frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial s}\delta s\right) \\ &= \left(\frac{\partial \widetilde{I}}{\partial u} - \Psi^T \frac{\partial R}{\partial u}\right)\delta u && + \left(\frac{\partial \widetilde{I}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}\right)\delta s\end{aligned}$$

$$\begin{aligned}\delta I &= \left(\frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial s}\delta s\right) && - \psi^T\left(\frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial s}\delta s\right)\\ &= \left(\frac{\partial \widetilde{I}}{\partial u} - \psi^T\frac{\partial R}{\partial u}\right)\delta u && + \left(\frac{\partial \widetilde{I}}{\partial s} - \psi^T\frac{\partial R}{\partial s}\right)\delta s\end{aligned}$$

Thus

$$\frac{DI}{Ds} = \frac{\partial \widetilde{I}}{\partial s} - \psi^T\frac{\partial R}{\partial s}$$

$$\text{if} \quad \left[\frac{\partial R}{\partial u}\right]^T \psi = \left[\frac{\partial \widetilde{I}}{\partial u}\right]^T .$$

Example: Discrete Adjoint for FEM

- FEM $\Rightarrow$ linear system

$$K(s)u = b(s) \qquad\qquad R(u,s) := K(s)u - b(s)$$

- Functional $I := [g(s)]^T u$
- discrete-adjoint equation

$$K(s)^T \Psi = g(s)$$

- Just one additional solve to get whole gradient, independent of $\dim(s)$.

Example: Discrete Adjoint for FEM

- FEM $\Rightarrow$ linear system

$$K(s)u = b(s) \qquad\qquad R(u,s) := K(s)u - b(s)$$

- Functional $I := [g(s)]^T u$
- discrete-adjoint equation

$$K(s)^T \Psi = g(s)$$

- Just one additional solve to get whole gradient, independent of $\dim(s)$.

Example: Discrete Adjoint for FEM

- FEM $\Rightarrow$ linear system

$$K(s)u = b(s) \qquad\qquad R(u,s) := K(s)u - b(s)$$

- Functional $I := [g(s)]^T u$
- discrete-adjoint equation

$$K(s)^T \Psi = g(s)$$

evaluation of the gradient

$$\frac{DI}{Ds} = \frac{\partial I}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

- Just **one** additional solve to get whole gradient, independent of $\dim(s)$.

Example: Discrete Adjoint for FEM

- FEM $\Rightarrow$ linear system

$$K(s)u = b(s) \qquad\qquad R(u,s) := K(s)u - b(s)$$

- Functional $I := [g(s)]^T u$
- discrete-adjoint equation

$$K(s)^T \Psi = g(s)$$

evaluation of the gradient

$$\frac{DI}{Ds} = \frac{\partial I}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

- Just **one** additional solve to get whole gradient, independent of dim(s).

Example for differentiating FE code

- Differentiate performance functional in FEM code wrt. domain geometry

solve

$$K(s)^T \Psi = g(s)$$

then evaluate gradient

$$\frac{DI}{Ds} = \frac{\partial I}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

- difficulty:
  require $\partial R/\partial s$, $\partial I/\partial s$, i.e. derivatives of the FE discretisation wrt. node positions

Example for differentiating FE code

- Differentiate performance functional in FEM code wrt. domain geometry

solve

$$K(s)^T \Psi = g(s)$$

then evaluate gradient

$$\frac{DI}{Ds} = \frac{\partial I}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

- difficulty:
  require $\partial R/\partial s$, $\partial I/\partial s$, i.e. derivatives of the FE discretisation wrt. node positions

Example for differentiating FE code

- Differentiate performance functional in FEM code wrt. domain geometry

solve

$$K(s)^T \Psi = g(s)$$

then evaluate gradient

$$\frac{DI}{Ds} = \frac{\partial I}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

- difficulty:
  require $\partial R/\partial s$, $\partial I/\partial s$, i.e. derivatives of the FE discretisation wrt. node positions

Example for $\partial R/\partial s$ for FE code

- model problem:

$$F(s) := \int_{T_\ell} \nabla\varphi_j(x) \cdot \nabla\varphi_i(x) \, \mathrm{d}\Omega \tag{2}$$

- unstructured mesh, isoparametric elements
- (2) is calculated by quadrature-formula on reference element

Example for $\partial R/\partial s$ for FE code

- model problem:

$$F(s) := \int_{T_\ell} \nabla\varphi_j(x) \cdot \nabla\varphi_i(x) \, \mathrm{d}\Omega \tag{2}$$

- unstructured mesh, isoparametric elements
- (2) is calculated by quadrature-formula on reference element

Example for $\partial R/\partial s$ for FE code

- model problem:

$$F(s) := \int_{T_\ell} \nabla\varphi_j(x) \cdot \nabla\varphi_i(x) \, \mathrm{d}\Omega \tag{2}$$

- unstructured mesh, isoparametric elements
- (2) is calculated by quadrature-formula on reference element

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$x = M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x})$$

$$J := \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T$$

$$\varphi(x) = \hat{\varphi}(M^{-1}(x))$$

$$\nabla_T \varphi_i(M(\hat{x}_k)) = J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k)$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$x = M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x})$$

$$J := \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T$$

$$\varphi(x) = \hat{\varphi}(M^{-1}(x))$$

$$\nabla_T \varphi_i(M(\hat{x}_k)) = J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k)$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$\begin{aligned} x =& M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x}) \\ J :=& \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T \\ \varphi(x) =& \hat{\varphi}(M^{-1}(x)) \\ \nabla_T \varphi_i(M(\hat{x}_k)) =& J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k) \end{aligned}$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$\begin{aligned} x =& M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x}) \\ J :=& \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T \\ \varphi(x) =& \hat{\varphi}(M^{-1}(x)) \\ \nabla_T \varphi_i(M(\hat{x}_k)) =& J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k) \end{aligned}$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$\begin{aligned} x =& M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x}) \\ J :=& \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T \\ \varphi(x) =& \hat{\varphi}(M^{-1}(x)) \\ \nabla_T \varphi_i(M(\hat{x}_k)) =& J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k) \end{aligned}$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$\begin{aligned} x =& M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x}) \\ J :=& \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T \\ \varphi(x) =& \hat{\varphi}(M^{-1}(x)) \\ \nabla_T \varphi_i(M(\hat{x}_k)) =& J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k) \end{aligned}$$

Example for $\partial R/\partial s$ for FE code (2)

$$F(s) = \sum_{k=1}^{m} \nabla_T \varphi_j(M(\hat{x}_k)) \cdot \nabla_T \varphi_i(M(\hat{x}_k)) |\det(J(\hat{x}_k))| w_k$$

$$\begin{aligned} x =& M(\hat{x}) = \sum_i s_i \hat{\varphi}_i(\hat{x}) \\ J :=& \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_i s_i \left[\hat{\nabla} \hat{\varphi}_i(\hat{x})\right]^T \\ \varphi(x) =& \hat{\varphi}(M^{-1}(x)) \\ \nabla_T \varphi_i(M(\hat{x}_k)) =& J^{-1} \hat{\nabla} \hat{\varphi}_i(\hat{x}_k) \end{aligned}$$

$\Rightarrow$ if derivatives of highlighted terms are calculated, only have to use product rule for rest

Example for $\partial R/\partial s$ for FE code (3)

Proposition 1

$$\frac{\partial[\nabla_T \varphi^{(i)}]_u}{\partial[s_k]_t} = -\left[\nabla_T \psi^{(k)}\right]_u \left[\nabla_T \varphi^{(i)}\right]_t$$

$$\frac{\partial|\det(J_T)|}{\partial s_{g_T(k)}} = |\det(J_T)| J_T^{-T} \hat{\nabla} \widehat{\psi}^{(k)}(\hat{x}_\ell).$$

Proof:
[S. PhD Thesis], [S./Jimack 2008]

- first part:
  implicit function theorem, re-organising terms
- second part:
  utilise adjoint representation of inverse of J_T

Example for $\partial R/\partial s$ for FE code (3)

Proposition 1

$$\frac{\partial[\nabla_T \varphi^{(i)}]_u}{\partial[s_k]_t} = -\left[\nabla_T \psi^{(k)}\right]_u \left[\nabla_T \varphi^{(i)}\right]_t$$

$$\frac{\partial|\det(J_T)|}{\partial s_{g_T(k)}} = |\det(J_T)| J_T^{-T} \hat{\nabla} \widehat{\psi}^{(k)}(\hat{x}_\ell).$$

Proof:
[S. PhD Thesis], [S./Jimack 2008]

- first part:
  implicit function theorem, re-organising terms
- second part:
  utilise adjoint representation of inverse of J_T

Optimisation solver

- discrete adjoint allows efficient gradient evaluation for objective function
- use SQP solver with BFGS update formula to evaluate Hessian
  - not just steepest descent!
- modular nature of the approach allows to use any optimisation solver
- we use (a MATLAB interface to) `DONLP2` by P. Spellucci in our examples

Optimisation solver

- discrete adjoint allows efficient gradient evaluation for objective function
- use SQP solver with BFGS update formula to evaluate Hessian
  - not just steepest descent!
- modular nature of the approach allows to use any optimisation solver
- we use (a MATLAB interface to) `DONLP2` by P. Spellucci in our examples

Optimisation solver

- discrete adjoint allows efficient gradient evaluation for objective function
- use SQP solver with BFGS update formula to evaluate Hessian
  - not just steepest descent!
- modular nature of the approach allows to use any optimisation solver
- we use (a MATLAB interface to) `DONLP2` by P. Spellucci in our examples

- stationary incompressible Navier-Stokes

$$-\nu\ \Delta \mathbf{u} + \mathbf{u}.\nabla \mathbf{u} + \nabla p = \mathbf{f},$$
$$\nabla.\mathbf{u} = 0.$$

- model for incompressible fluids or compressible fluids at low Mach number (< 0.3).
- existence and uniqueness for small data, e.g. [Gunzburger 1989]

Navier-Stokes solver

`FEINS`:

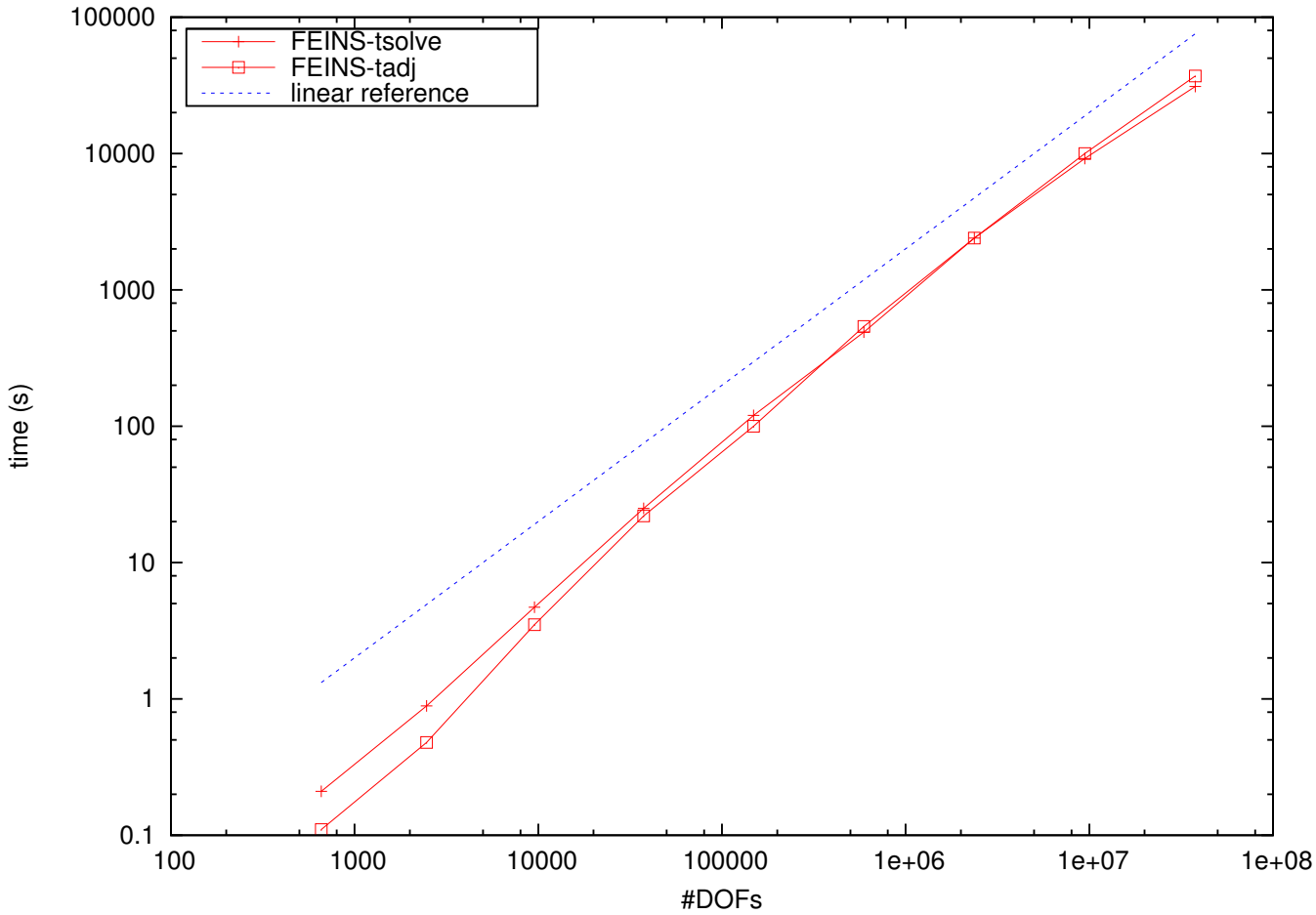
- Taylor-Hood finite elements ($P_2 - P_1$), unstructured meshes of triangles
- full-multigrid and Newton-method for nonlinear algebraic system
- GMRES preconditioned with F_p preconditioner, see e.g. [Elmann/Silvester/Wathen 2005].

$$K = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}$$

$$C^{-1} = \begin{bmatrix} I & 0 \\ 0 & M_p^{-1} F_p \, A_p^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}$$

- multigrid preconditioned GMRES for F^{-1} part

Navier-Stokes solver



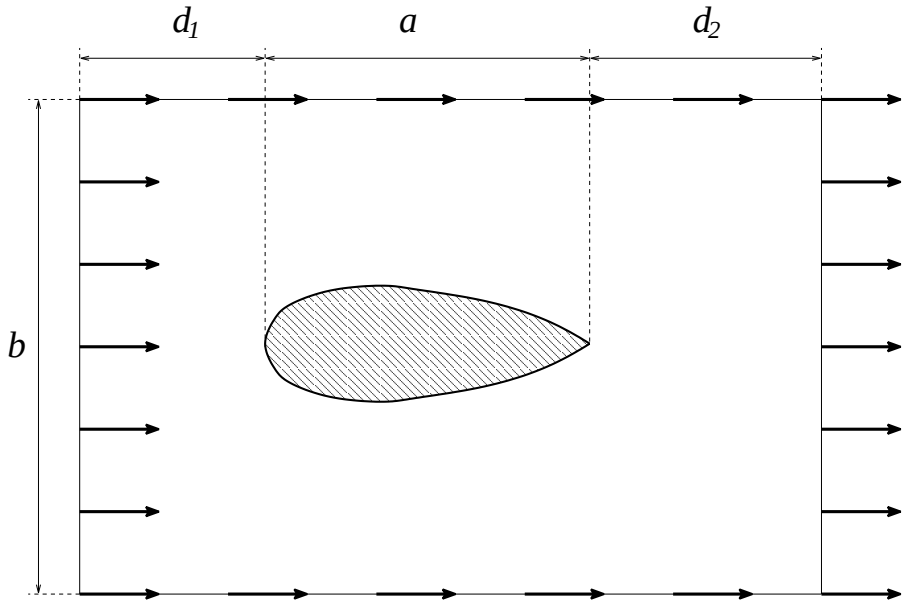
Example: Obstacle in channel

- An object is moving with $v = 1$ in a channel of viscous fluid.
- Goal: find shape such that drag $\rightarrow$ min.

$$F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \int_{\partial Q} \mu(\nabla u + \nabla u^T)\, n \, \mathrm{d}\Gamma - \int_{\partial Q} p\, n \, \mathrm{d}\Gamma,$$

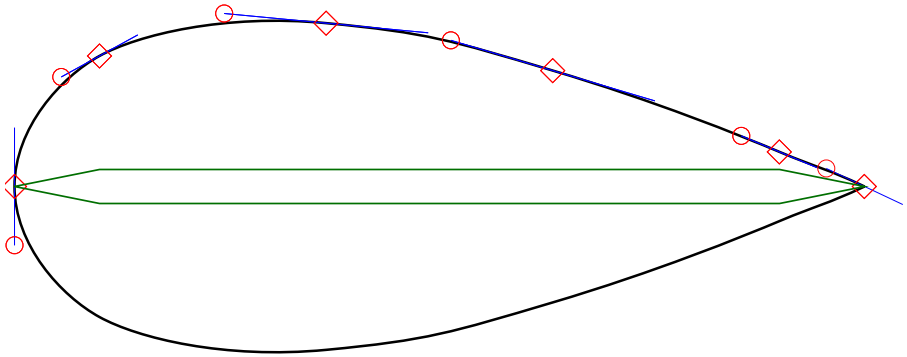
$$I := F_1.$$

- constraint: $a = const$, $V \geq const.$, symmetric



Shape discretisation

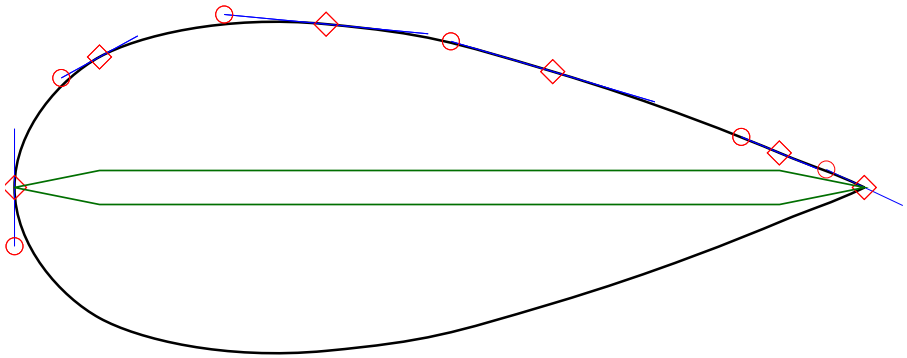
- Shape discretised by Bezier-Splines:



- use mesh generator `triangle` [Shewchuk 2002] to generate coarse meshes from boundary mesh
- the fine meshes are deformed to account for shape changes and curved geometry

Shape discretisation

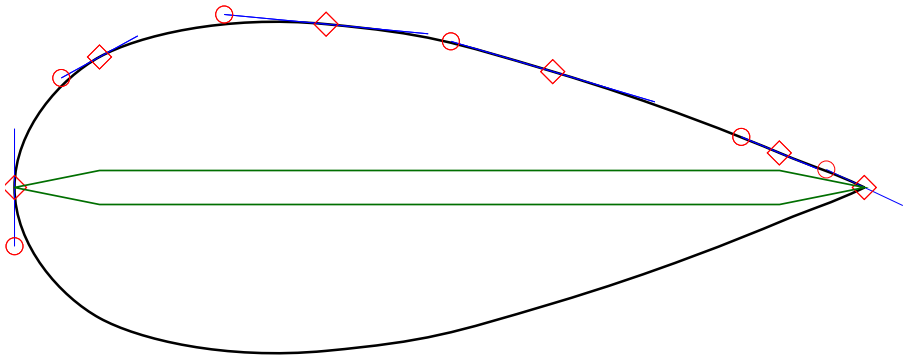
- Shape discretised by Bezier-Splines:



- use mesh generator `triangle` [Shewchuk 2002] to generate coarse meshes from boundary mesh
- the fine meshes are deformed to account for shape changes and curved geometry

Shape discretisation

- Shape discretised by Bezier-Splines:



- use mesh generator `triangle` [Shewchuk 2002] to generate coarse meshes from boundary mesh
- the fine meshes are deformed to account for shape changes and curved geometry

Example 2: Obstacle in channel

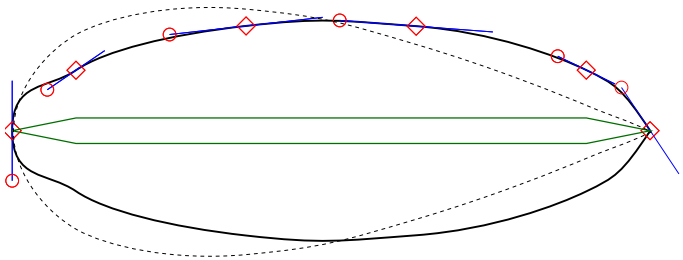
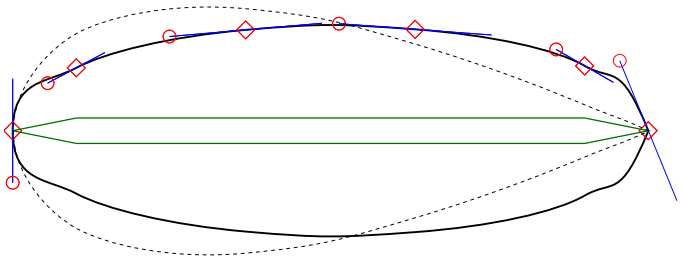
- Optimisation software: `DONLP2` [P. Spellucci]
- uses SQP with BFGS-update formula for the Hessian.
- Here:
  - 10 parameters to optimise
  - 16 SQP steps
  - 60 evaluations in total
  - $\sim$ 575,000 DOFs in each nonlinear equation system, $I(s)$: 65 min, adjoint 43 min

Example 2: Obstacle in channel

- Optimisation software: `DONLP2` [P. Spellucci]
- uses SQP with BFGS-update formula for the Hessian.
- Here:
  - 10 parameters to optimise
  - 16 SQP steps
  - 60 evaluations in total
  - $\sim$ 575,000 DOFs in each nonlinear equation system, $I(s)$: 65 min, adjoint 43 min

Example 2: Obstacle in channel

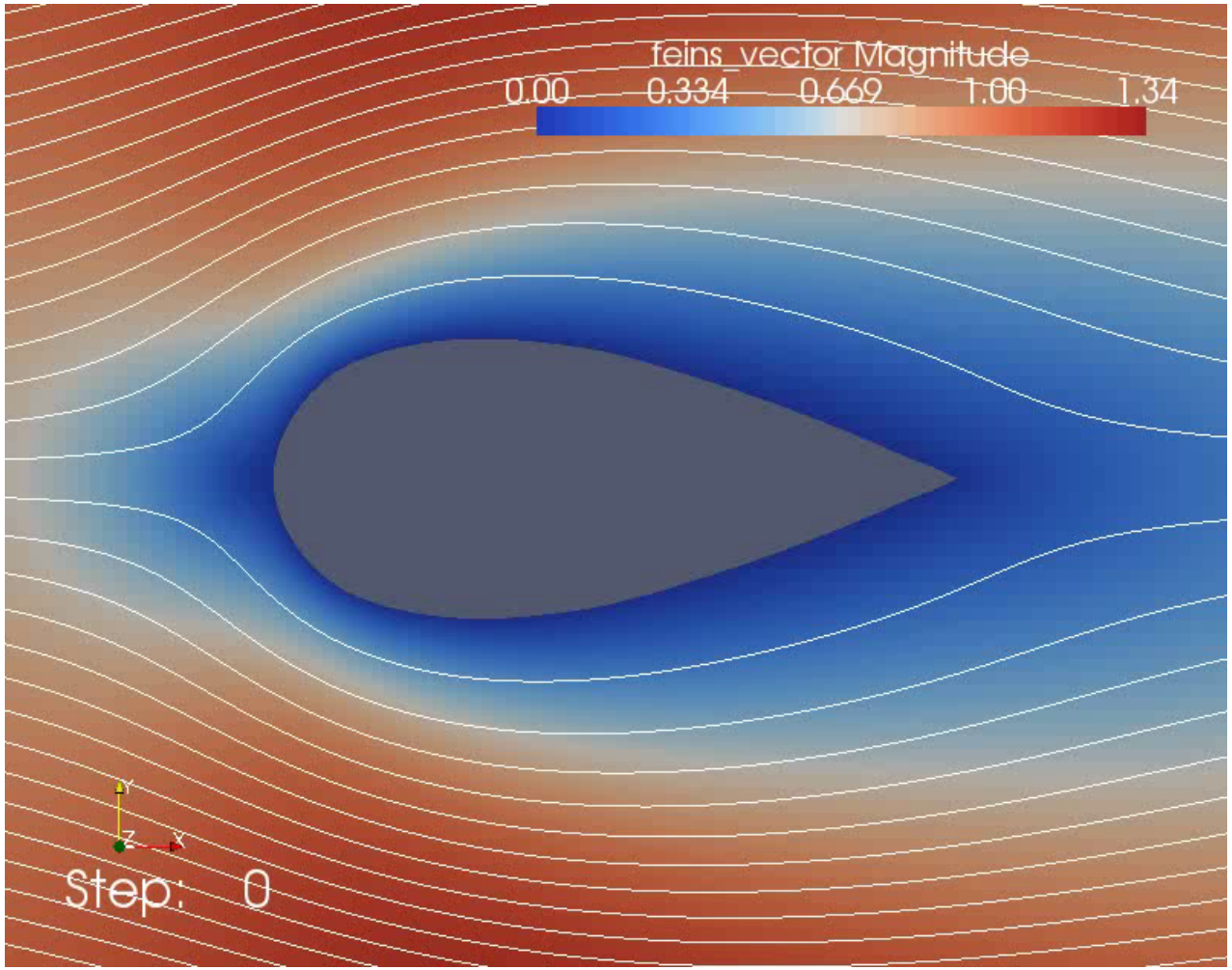
optimisation results

	$Re = 10$	$Re = 20$
		
I_{ini}	1.4056	0.8743
I_{opt}	1.3714	0.8452
#it	18	16
#evals	69	60
#DOFs	$\approx 145,000$	$\approx 575,000$

Example 2: Obstacle in channel

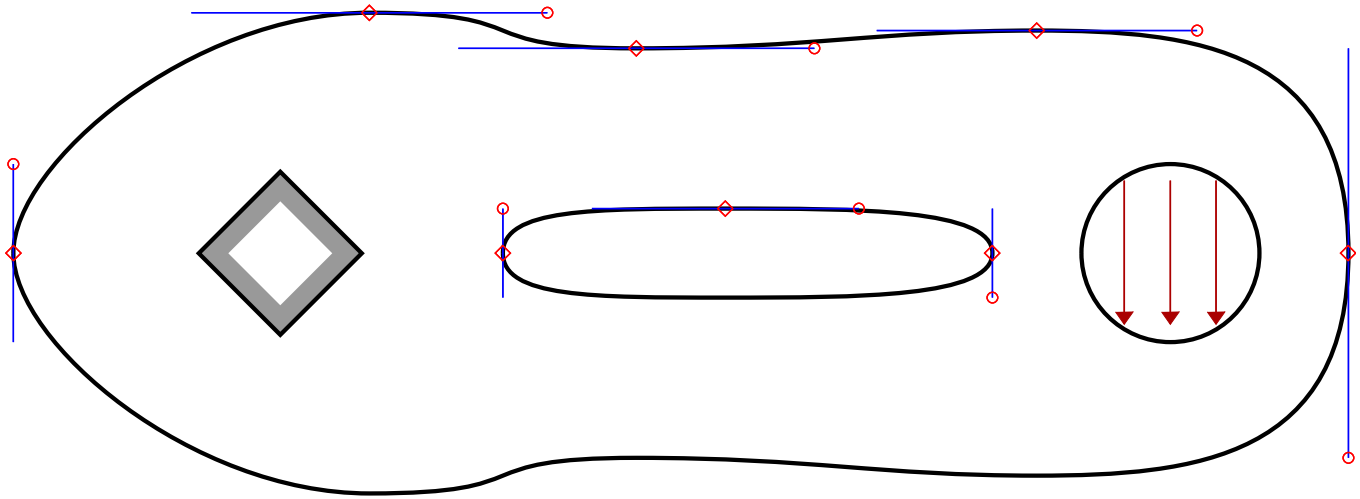
optimisation movie

$f_0 = 1.4056 \qquad f_* = 1.3714$



Example 3: linear elasticity
(with Andreas Günnel)

sketch of pedal crank problem:



24 free parameters

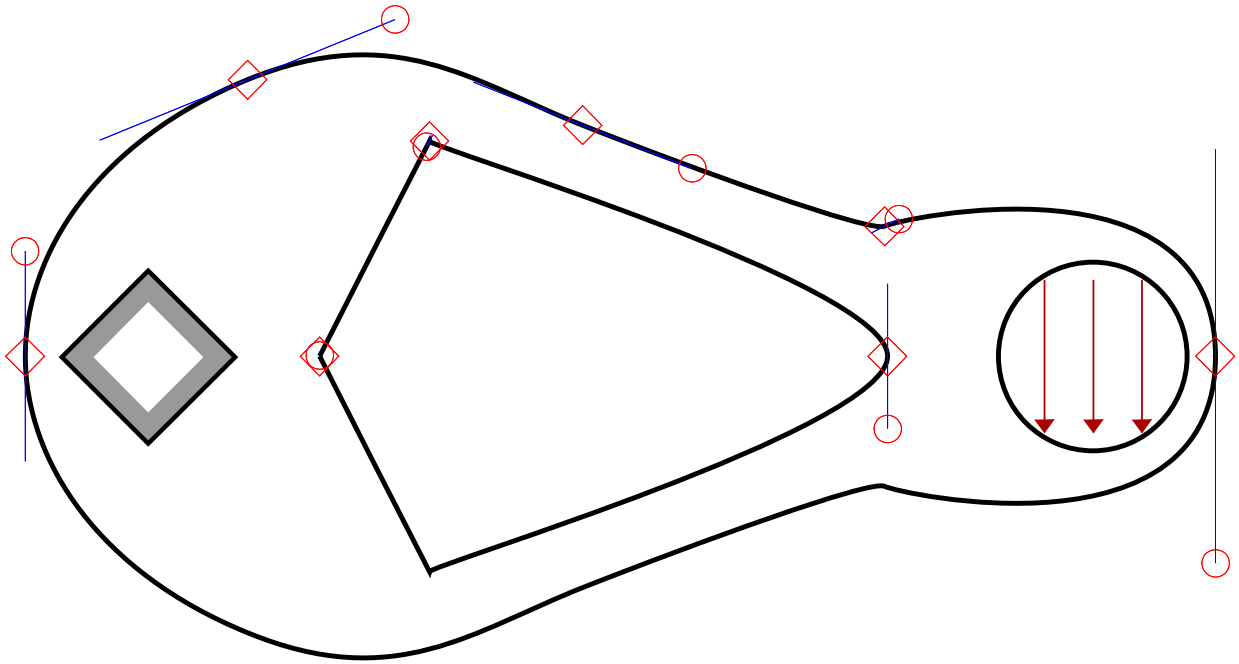
performance functional: deformation energy

$$I(\mathcal{F}) := \frac{1}{2}a(u,u) - b(u) \quad + \alpha \int_\Omega 1 \, \mathrm{d}\Omega$$

adaptive discretisation (residual error estimate)

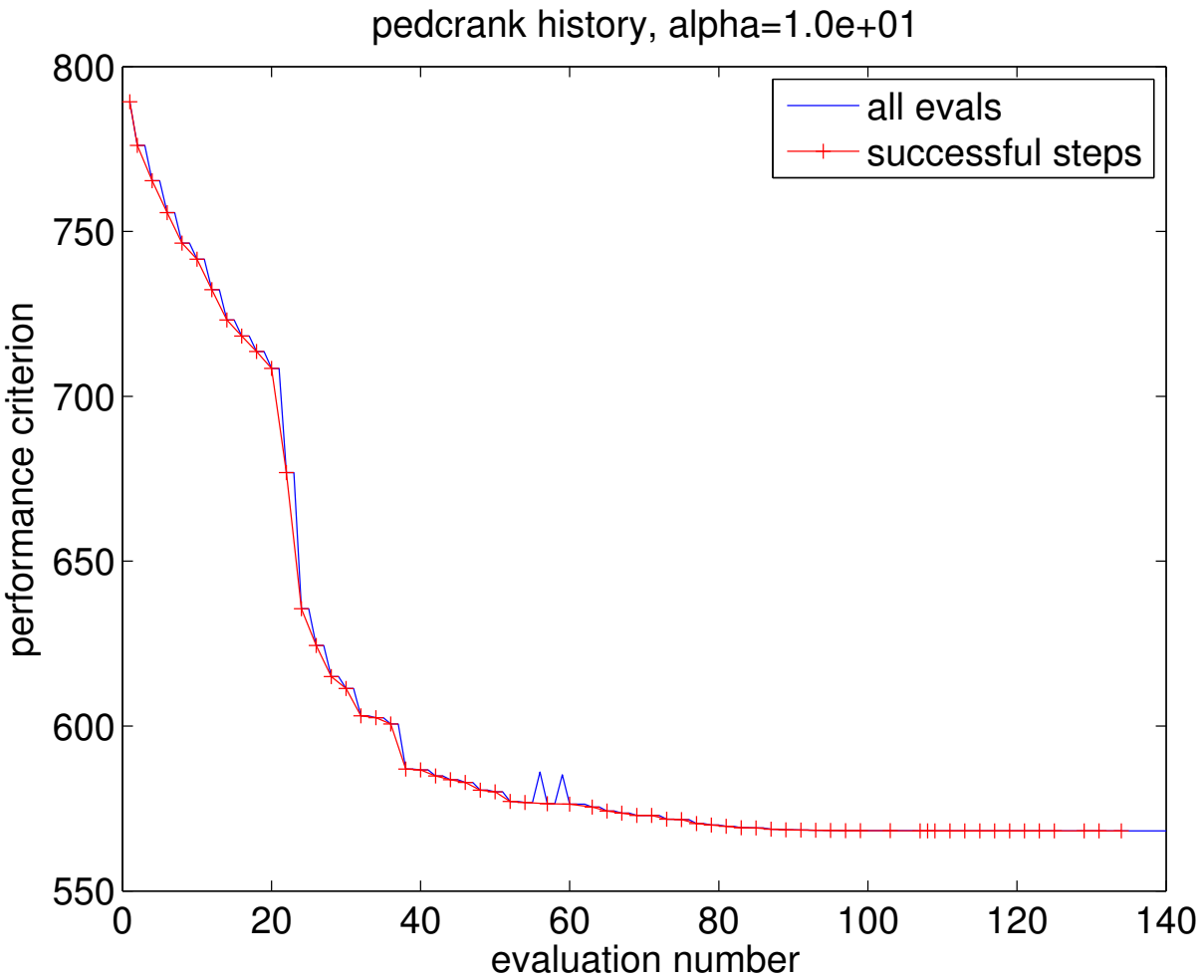
Example 3: linear elasticity
(with Andreas Günnel)

optimal shape: $I(\mathcal{F})$ $\alpha = 10$

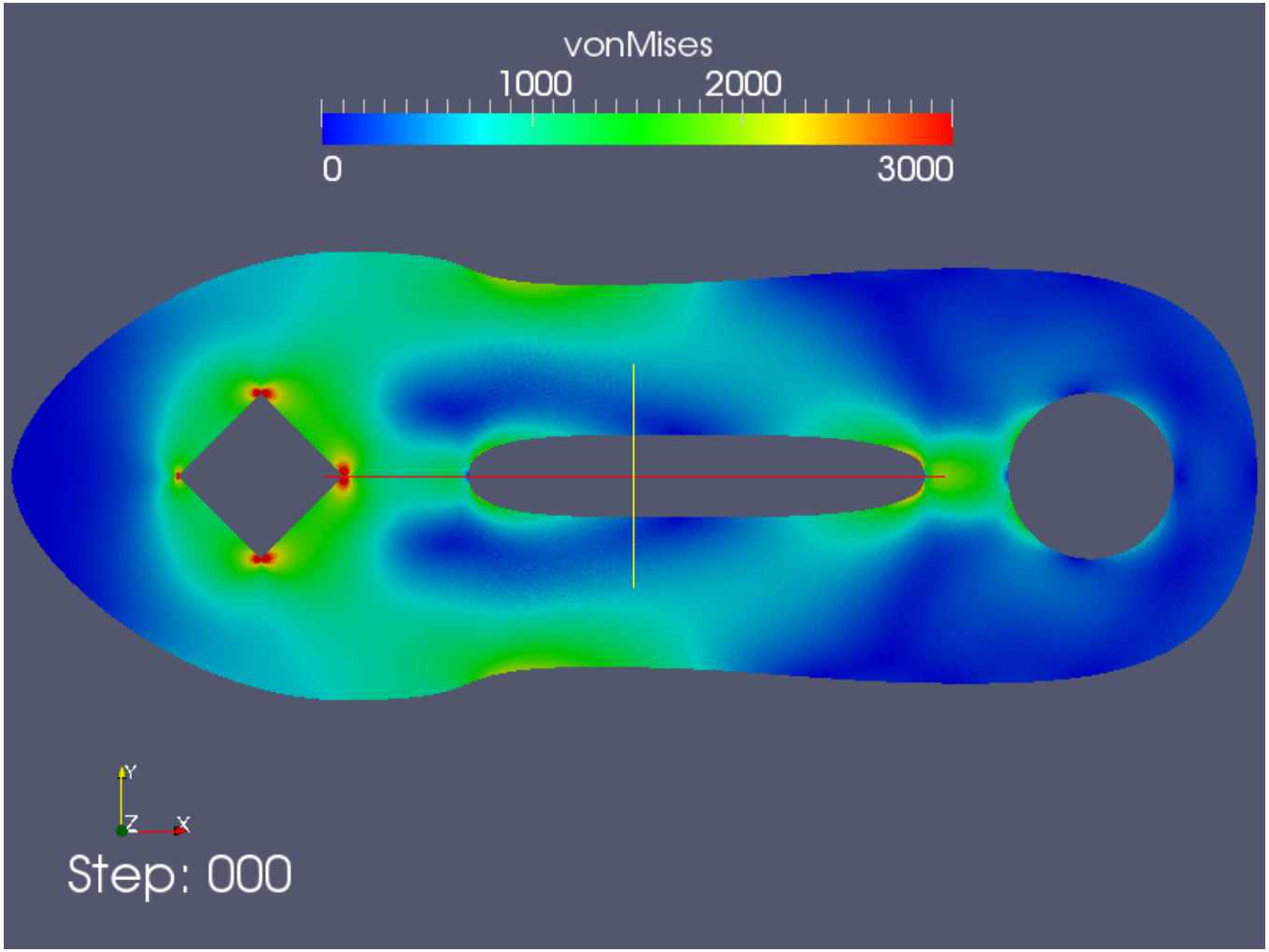


Example 3: linear elasticity
(with Andreas Günnel)

convergence history of $I(\mathcal{F})$ $\alpha = 10$



Example 3: linear elasticity
(with Andreas Günnel)



Conclusions

- Discrete adjoint technique very general, also for other PDE optimisation problems.
- Examples:
  - anisotropic mesh adaptation by optimisation of node positions,
  - Navier-Stokes with FVM discretisation.
- FEM solver `FEINS` efficient
  - Navier-Stokes only for low Re
  - linear elasticity
- Demonstrated shape optimisation feasible for a set of model problems

Thank you!