# From Simulation to Optimization: Discrete Adjoint Equations

René Schneider

Mathematik in Industrie und Technik
Fakultät für Mathematik
TU Chemnitz

Magdeburg, 5. November 2010

Department of Mathematics

CHEMNITZ UNIVERSITY OF TECHNOLOGY 175 Years

# Overview

1. Motivation
2. Sensitivity analysis
3. Examples

# Motivation

Modelling and Simulation of scientific/engineering problems

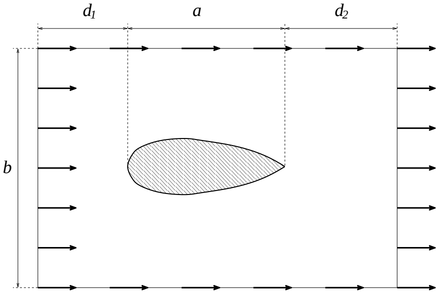
- modelling
- PDE or ODE
- numerical approximation/simulation
- predictions, analysis
- really only f(x) for optimisation

# Motivation

Modelling and Simulation of scientific/engineering problems

- modelling
- PDE or ODE
- numerical approximation/simulation
- predictions, analysis
- really only f(x) for optimisation

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



# What we want

- Evaluation of $J(s)$, $\frac{DJ}{Ds} := \text{grad}\, J(s)$ (sensitivity analysis)
- use standard gradient optimisation techniques

# What we want

- Evaluation of $J(s)$, $\frac{\mathrm{D}J}{\mathrm{D}s} := \operatorname{grad} J(s)$ (sensitivity analysis)
- use standard gradient optimisation techniques
  - NOT just steepest descent!
  - quasi Newton-type (SQP with BFGS for Hessian)
  - robust and reliable algorithms and software available, e.g. [DONLP2] by P. Spellucci.

- Evaluation of $J(s)$, $\frac{\mathrm{D}J}{\mathrm{D}s} := \operatorname{grad} J(s)$ (sensitivity analysis)
- use standard gradient optimisation techniques
  - NOT just steepest descent!
  - quasi Newton-type (SQP with BFGS for Hessian)
  - robust and reliable algorithms and software available, e.g. [DONLP2] by P. Spellucci.

# What we want

- Evaluation of $J(s)$, $\frac{\mathrm{D}J}{\mathrm{D}s} := \operatorname{grad} J(s)$ (sensitivity analysis)
- use standard gradient optimisation techniques
  - NOT just steepest descent!
  - quasi Newton-type (SQP with BFGS for Hessian)
  - robust and reliable algorithms and software available, e.g. [DONLP2] by P. Spellucci.

# What we want

- Evaluation of $J(s)$, $\frac{\mathrm{D}J}{\mathrm{D}s} := \mathrm{grad}\, J(s)$ (sensitivity analysis)
- use standard gradient optimisation techniques
  - NOT just steepest descent!
  - quasi Newton-type (SQP with BFGS for Hessian)
  - robust and reliable algorithms and software available, e.g. [DONLP2] by P. Spellucci.

# Assumptions

- number of parameters to be optimised $\gg 1$
- simulation expensive compared to post-processing
- Let $J$ be a scalar valued function

$$J(s) = \widetilde{J}(\underline{u}(s), s), \quad \text{with vector } \underline{u}(s) \text{ defined by}$$
$$0 = R(\underline{u}(s), s).$$

# Assumptions

- number of parameters to be optimised $\gg 1$
- simulation expensive compared to post-processing
- Let $J$ be a scalar valued function

$$\begin{aligned} J(s) &= \widetilde{J}(\underline{u}(s), s), \quad \text{with vector } \underline{u}(s) \text{ defined by} \\ 0 &= R(\underline{u}(s), s). \end{aligned}$$

# Naive way 1: Finite Difference

$$J(s) = \widetilde{J}(\underline{u}(s), s)$$
$$0 = R(\underline{u}(s), s)$$

solve $R(\underline{u}, s) = 0$, evaluate $J_0 := \widetilde{J}(\underline{u}, s)$
for $i = 1, \ldots, \dim(s)$
  perturb $i$-th parameter in $s$ by $h > 0$, $\tilde{s} = s + h$
  solve $R(\underline{\tilde{u}}, \tilde{s}) = 0$, evaluate $J_i := \widetilde{J}(\underline{\tilde{u}}, \tilde{s})$
  $\frac{\mathrm{D}J}{\mathrm{D}s_i} \approx \frac{J_i - J_0}{h}$

- Problems: inaccuracies, choice of $h$.

# Naive way 1: Finite Difference

$$J(s) = \widetilde{J}(\underline{u}(s), s)$$
$$0 = R(\underline{u}(s), s)$$

solve $R(\underline{u}, s) = 0$, evaluate $J_0 := \widetilde{J}(\underline{u}, s)$
for $i = 1, \ldots, \dim(s)$

perturb $i$-th parameter in $s$ by $h > 0$, $\tilde{s} = s + h$

solve $R(\tilde{\underline{u}}, \tilde{s}) = 0$, evaluate $J_i := \widetilde{J}(\tilde{\underline{u}}, \tilde{s})$

$\frac{\mathrm{D}J}{\mathrm{D}s_i} \approx \frac{J_i - J_0}{h}$

- Problems: inaccuracies, choice of $h$.

# Naive way 2: Sensitivity equation

$$J(s) = \widetilde{J}(\underline{u}(s), s)$$
$$0 = R(\underline{u}(s), s)$$

for $\delta s = i$-th unit vector, $i = 1, \ldots, \dim(s)$,

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

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

$$\frac{\mathrm{D}J}{\mathrm{D}s_i} = \delta J$$

# Naive way 2: Sensitivity equation

$$J(s) = \widetilde{J}(\underline{u}(s), s)$$
$$0 = R(\underline{u}(s), s)$$

for $\delta s = i$-th unit vector, $i = 1, \ldots, \dim(s)$,

$$0 = \delta R = \frac{\partial R}{\partial \underline{u}} \delta \underline{u} + \frac{\partial R}{\partial s} \delta s$$
$$\delta J = \frac{\partial \widetilde{J}}{\partial \underline{u}} \delta \underline{u} + \frac{\partial \widetilde{J}}{\partial s} \delta s$$
$$\frac{\mathrm{D}J}{\mathrm{D}s_i} = \delta J$$

# Naive way 2: Sensitivity equation

$$
\begin{aligned}
J(s) &= \widetilde{J}(\underline{u}(s), s) \\
0 &= R(\underline{u}(s), s)
\end{aligned}
$$

for $\delta s = i$-th unit vector, $i = 1, \ldots, \dim(s)$,

$$
\begin{aligned}
0 = \delta R &= \frac{\partial R}{\partial \underline{u}} \delta \underline{u} + \frac{\partial R}{\partial s} \delta s \\
\delta J &= \frac{\partial \widetilde{J}}{\partial \underline{u}} \delta \underline{u} + \frac{\partial \widetilde{J}}{\partial s} \delta s \\
\frac{\mathrm{D}J}{\mathrm{D}s_i} &= \delta J
\end{aligned}
$$

# Way 3: Discrete adjoint equation

Take

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

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

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

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

# Way 3: Discrete adjoint equation

Take

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

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

# Way 3: Discrete adjoint equation

Take

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

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

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

Thus $\delta J$ can be evaluated without knowing $\delta \underline{u}$ if

$$\left[ \frac{\partial R}{\partial \underline{u}} \right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

# Way 3: Discrete adjoint equation

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

Thus $\delta J$ can be evaluated without knowing $\delta \underline{u}$ if

$$\left[ \frac{\partial R}{\partial \underline{u}} \right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

Then

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

# Way 3: Discrete adjoint equation

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

Thus $\delta J$ can be evaluated without knowing $\delta \underline{u}$ if

$$\left[ \frac{\partial R}{\partial \underline{u}} \right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

Then

$$\frac{\mathrm{D} J}{\mathrm{D} s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

# Way 3: Discrete adjoint equation

$$\left[\frac{\partial R}{\partial \underline{u}}\right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

Discussion:

- $0 = R(\underline{u}, s)$ is $N$ equations in $N$ unknowns $\underline{u}$, then the adjoint is $N$ equations in $N$ unknowns $\Psi$.

Why important:

$$\left[\frac{\partial R}{\partial \underline{u}}\right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

Discussion:

- $0 = R(\underline{u}, s)$ is $N$ equations in $N$ unknowns $\underline{u}$, then the adjoint is $N$ equations in $N$ unknowns $\Psi$.
- If (nonlinear) system $0 = R(\underline{u}, s)$ is uniquely solvable, then adjoint is a uniquely solvable linear system.

Why important:

# Way 3: Discrete adjoint equation

$$\left[\frac{\partial R}{\partial \underline{u}}\right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

Discussion:

- $0 = R(\underline{u}, s)$ is $N$ equations in $N$ unknowns $\underline{u}$, then the adjoint is $N$ equations in $N$ unknowns $\Psi$.
- If (nonlinear) system $0 = R(\underline{u}, s)$ is uniquely solvable, then adjoint is a uniquely solvable linear system.

Why important:

- Requires only one solution of adjoint problem, independent of $\dim(s)$.

# Way 3: Discrete adjoint equation

$$\left[\frac{\partial R}{\partial \underline{u}}\right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

Discussion:

- $0 = R(\underline{u}, s)$ is $N$ equations in $N$ unknowns $\underline{u}$, then the adjoint is $N$ equations in $N$ unknowns $\Psi$.
- If (nonlinear) system $0 = R(\underline{u}, s)$ is uniquely solvable, then adjoint is a uniquely solvable linear system.

Why important:

- Requires only one solution of adjoint problem, independent of $\dim(s)$.
- In contrast: sensitivity equation or finite differences require one solution of PDE per component of $s$.

# Way 3: Discrete adjoint equation

$$\left[\frac{\partial R}{\partial \underline{u}}\right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

Discussion:

- $0 = R(\underline{u}, s)$ is $N$ equations in $N$ unknowns $\underline{u}$, then the adjoint is $N$ equations in $N$ unknowns $\Psi$.
- If (nonlinear) system $0 = R(\underline{u}, s)$ is uniquely solvable, then adjoint is a uniquely solvable linear system.

Why important:

- Requires only one solution of adjoint problem, independent of $\dim(s)$.
- In contrast: sensitivity equation or finite differences require one solution of PDE per component of $s$.

# Example 1: Heat conduction

- Stationary heat equation

$$-\Delta u = f \qquad \text{in } \Omega$$
$$u = 0 \qquad \text{on } \partial\Omega$$

- Finite Element or Finite Difference Discretisation

$$\Rightarrow K(s)\underline{u} = b(s)$$
$$\Leftrightarrow 0 = R(\underline{u}, s) := K(s)\underline{u} - b(s)$$

# Example 1: Heat conduction

- Stationary heat equation

$$-\Delta u = f \qquad \text{in } \Omega$$
$$u = 0 \qquad \text{on } \partial\Omega$$

- Finite Element or Finite Difference Discretisation

$$\Rightarrow K(s)\underline{u} = b(s)$$
$$\Leftrightarrow 0 = R(\underline{u}, s) := K(s)\underline{u} - b(s)$$

$$\frac{\partial R(\underline{u}, s)}{\partial \underline{u}} = K$$

Adjoint FEM

$$\Rightarrow K^T \Psi = \frac{\partial J}{\partial \underline{u}}$$

# Example 1: Heat conduction

- Stationary heat equation

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega \end{aligned}$$

- Finite Element or Finite Difference Discretisation

$$\Rightarrow K(s)\underline{u} = b(s)$$
$$\Leftrightarrow 0 = R(\underline{u}, s) := K(s)\underline{u} - b(s)$$

$$\frac{\partial R(\underline{u}, s)}{\partial \underline{u}} = K$$

Adjoint FEM

$$\Rightarrow K^T \Psi = \frac{\partial J}{\partial \underline{u}}$$

# Example 2: ODE solver

- e.g. forward Euler

$$\begin{aligned} \dot{u} &= f(u, t, s) && \forall t \in [a, b], \\ u(a) &= u_a(s) \end{aligned}$$

$$\begin{aligned} \underline{u}_0 &= u_a(s) \\ \underline{u}_i &= \underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s) && \forall i = 1, \dots, n \end{aligned}$$

$$\Rightarrow 0 = R_i(\underline{u}, s) = \underline{u}_i - \big(\underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s)\big)$$

$$\frac{\partial R(\underline{u}, s)}{\partial \underline{u}} = \begin{bmatrix} I & & & & \\ \delta_1 & I & & & \\ & \delta_2 & I & & \\ & & \ddots & \ddots & \\ & & & \delta_n & I \end{bmatrix},$$

$$\delta_i = -\left(I + \tau \frac{\partial f(u_{i-1}, t_{i-1})}{\partial u}\right)$$

# Example 2: ODE solver

- e.g. forward Euler

$$
\begin{aligned}
\dot{u} &= f(u,t,s) && \forall t \in [a,b], \\
u(a) &= u_a(s)
\end{aligned}
$$

$$
\begin{aligned}
\underline{u}_0 &= u_a(s) \\
\underline{u}_i &= \underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s) && \forall i = 1, \dots, n
\end{aligned}
$$

$$
\Rightarrow 0 = R_i(\underline{u}, s) = \underline{u}_i - \left(\underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s)\right)
$$

$$
\frac{\partial R(\underline{u}, s)}{\partial \underline{u}}^{T} = \begin{bmatrix} I & \delta_1^T & & & \\ \delta_1 & I & \delta_2^T & & \\ & \delta_2 & I & \ddots & \\ & & \ddots & \ddots & \delta_n^T \\ & & & \delta_n & I \end{bmatrix},
$$

$$
\delta_i = -\left( I + \tau \frac{\partial f(u_{i-1}, t_{i-1})}{\partial u} \right)
$$

# Example 2: ODE solver

- e.g. forward Euler

$$\begin{aligned} \dot{u} &= f(u,t,s) && \forall t \in [a,b], \\ u(a) &= u_a(s) \end{aligned}$$

$$\begin{aligned} \underline{u}_0 &= u_a(s) \\ \underline{u}_i &= \underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s) && \forall i = 1,\dots,n \end{aligned}$$

$$\Rightarrow 0 = R_i(\underline{u}, s) = \underline{u}_i - \left(\underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s)\right)$$

$$\frac{\partial R(\underline{u},s)}{\partial \underline{u}}^T = \begin{bmatrix} I & \delta_1^T & & & \\ \delta_1 & I & \delta_2^T & & \\ & \delta_2 & I & \ddots & \\ & & \ddots & \ddots & \delta_n^T \\ & & & \delta_n & I \end{bmatrix},$$

$$\delta_i = -\left(I + \tau \frac{\partial f(u_{i-1}, t_{i-1})}{\partial u}\right)$$

# Example 2: ODE solver

$$\frac{\partial R(\underline{u}, s)}{\partial \underline{u}}^T = \begin{bmatrix} I & \delta_1^T & & & \\ & I & \delta_2^T & & \\ & & I & \ddots & \\ & & & \ddots & \delta_n^T \\ & & & & I \end{bmatrix}$$

Adjoint discrete ODE:

$$\begin{aligned} \frac{\partial R}{\partial \underline{u}}^T \Psi &= \frac{\partial J}{\partial \underline{u}} \\ \Rightarrow \Psi_n &= \frac{\partial J}{\partial \underline{u}_n} \\ \Psi_i &= \Psi_{i+1} + \tau \frac{\partial f(\underline{u}_{i+1}, t_{i+1}, s)}{\partial u} \Psi_{i+1} + \frac{\partial J}{\partial \underline{u}_i} \quad \forall i = n-1, \ldots, 0 \end{aligned}$$

# Example 2: ODE solver

Compare:

Discrete ODE (forward in time)

$$\underline{u}_0 = u_a(s)$$
$$\underline{u}_i = \underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s) \qquad \forall i = 1, \dots, n$$

Discrete adjoint ODE (backward in time)

$$\Psi_n = \frac{\partial J}{\partial \underline{u}_n}$$
$$\Psi_i = \Psi_{i+1} + \tau \frac{\partial f(\underline{u}_{i+1}, t_{i+1}, s)}{\partial u} \Psi_{i+1} + \frac{\partial J}{\partial \underline{u}_i} \qquad \forall i = n-1, \dots, 0$$

# Example 2: ODE solver

Compare:

Discrete ODE (forward in time)

$$\underline{u}_0 = u_a(s)$$
$$\underline{u}_i = \underline{u}_{i-1} + \tau f(\underline{u}_{i-1}, t_{i-1}, s) \qquad \forall i = 1, \ldots, n$$

Discrete adjoint ODE (backward in time)

$$\Psi_n = \frac{\partial J}{\partial \underline{u}_n}$$
$$\Psi_i = \Psi_{i+1} + \tau \frac{\partial f(\underline{u}_{i+1}, t_{i+1}, s)}{\partial u} \Psi_{i+1} + \frac{\partial J}{\partial \underline{u}_i} \qquad \forall i = n-1, \ldots, 0$$

# Comparison of approaches

$$m := \dim(J), \quad k := \dim(s)$$

| sensitivity | discrete adj. | adj. PDE |
|---|---|---|
| $\mathcal{O}(k)$ LS | $\mathcal{O}(m)$ LS | $\mathcal{O}(m)$ PDE |
| discrete consistent | discrete consistent | not d. consistent |
| fwd adaptive | fwd adaptive | fwd+adj adaptive |
| fwd time | fwd+bwd time | fwd+bwd time |
| simplest | simple | more difficult (e.g. BC) |

Difficulties:

- A priori effort.
- Re-use of code.
- Verifiability of results.
- Automatisation of code generation.

Introduction: [Giles 2000]

# Comparison of approaches

$$m := \dim(J), \quad k := \dim(s)$$

| sensitivity | discrete adj. | adj. PDE |
|---|---|---|
| $\mathcal{O}(k)$ LS | $\mathcal{O}(m)$ LS | $\mathcal{O}(m)$ PDE |
| discrete consistent | discrete consistent | not d. consistent |
| fwd adaptive | fwd adaptive | fwd+adj adaptive |
| fwd time | fwd+bwd time | fwd+bwd time |
| simplest | simple | more difficult (e.g. BC) |

Difficulties:

- A priori effort.
- Re-use of code.
- Verifiability of results.
- Automatisation of code generation.

Introduction: [Giles 2000]

# Comparison of approaches

$$m := \dim(J), \quad k := \dim(s)$$

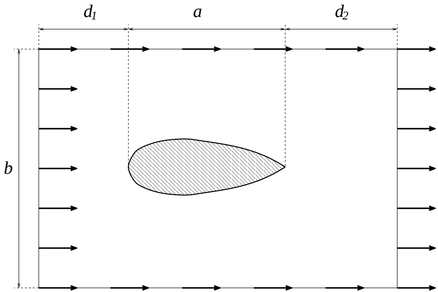
| sensitivity | discrete adj. | adj. PDE |
|---|---|---|
| $\mathcal{O}(k)$ LS | $\mathcal{O}(m)$ LS | $\mathcal{O}(m)$ PDE |
| discrete consistent | discrete consistent | not d. consistent |
| fwd adaptive | fwd adaptive | fwd+adj adaptive |
| fwd time | fwd+bwd time | fwd+bwd time |
| simplest | simple | more difficult (e.g. BC) |

Difficulties:

- A priori effort.
- Re-use of code.
- Verifiability of results.
- Automatisation of code generation.

Introduction: [Giles 2000]

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



# Example 3: Shape optimisation
## Navier Stokes, drag minimisation

- Fluid dynamics: stationary Navier-Stokes Equations

$$-\frac{1}{Re}\,\Delta\mathbf{u} + \mathbf{u}.\nabla\mathbf{u} + \nabla p = \mathbf{f},$$
$$\nabla.\mathbf{u} = 0.$$

- Discretised by FEM (Taylor-Hood elements).
- Shape discretised by Bezier-Splines:

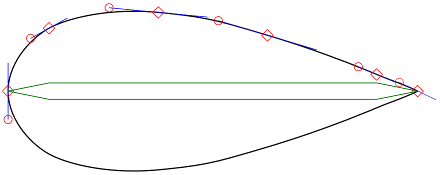
# Example 3: Shape optimisation

## Navier Stokes, drag minimisation

- Fluid dynamics: stationary Navier-Stokes Equations

$$-\frac{1}{Re}\Delta \mathbf{u} + \mathbf{u}.\nabla \mathbf{u} + \nabla p = \mathbf{f},$$
$$\nabla.\mathbf{u} = 0.$$

- Discretised by FEM (Taylor-Hood elements).
- Shape discretised by Bezier-Splines:



# Way 3: Discrete adjoint equation

$$\left[\frac{\partial R}{\partial \underline{u}}\right]^T \Psi = \frac{\partial \widetilde{J}}{\partial \underline{u}}$$

$$\frac{\mathrm{D}J}{\mathrm{D}s} = \frac{\partial \widetilde{J}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}$$

Discussion:

- $0 = R(\underline{u}, s)$ is $N$ equations in $N$ unknowns $\underline{u}$, then the adjoint is $N$ equations in $N$ unknowns $\Psi$.

Why important:

- Non-trivial part is $-\Psi^T \frac{\partial R}{\partial s}$, $s$ are nodal coordinates.
- Jacobian $\frac{\partial R}{\partial s}$ is sparse.
  But needs not be build as a whole.
- Only required for one matrix-vector product $-\Psi^T \frac{\partial R}{\partial s}$.
  $\Rightarrow$ Cheaper to evaluate only locally and sum up local contributions to $-\Psi^T \frac{\partial R}{\partial s}$ (assembly).
- Local contributions:
  by hand, algorithmic differentiation, or even finite differences [S./Jimack 2008]

# Evaluation of required partial derivatives

- Non-trivial part is $-\Psi^T \frac{\partial R}{\partial s}$, $s$ are nodal coordinates.
- Jacobian $\frac{\partial R}{\partial s}$ is sparse.
  But needs not be build as a whole.
- Only required for one matrix-vector product $-\Psi^T \frac{\partial R}{\partial s}$.
  $\Rightarrow$ Cheaper to evaluate only locally and sum up local contributions to $-\Psi^T \frac{\partial R}{\partial s}$ (assembly).
- Local contributions:
  by hand, algorithmic differentiation, or even finite differences
  [S./Jimack 2008]

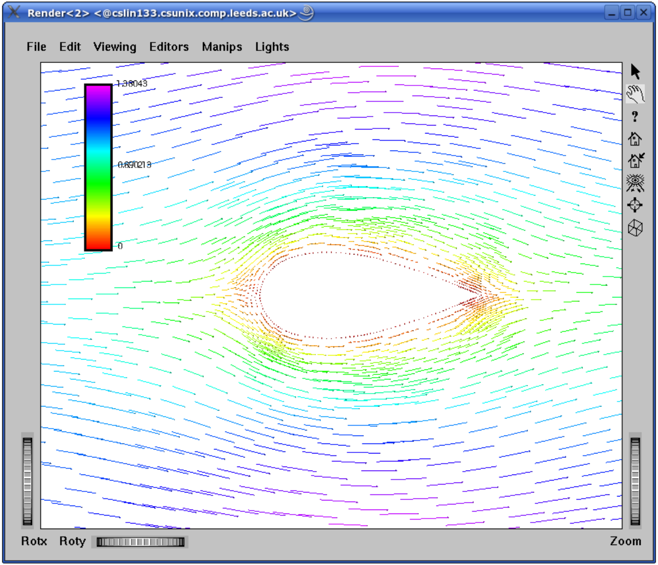
# Evaluation of required partial derivatives

- Non-trivial part is $-\Psi^T \frac{\partial R}{\partial s}$, $s$ are nodal coordinates.
- Jacobian $\frac{\partial R}{\partial s}$ is sparse.
  But needs not be build as a whole.
- Only required for one matrix-vector product $-\Psi^T \frac{\partial R}{\partial s}$.
  $\Rightarrow$ Cheaper to evaluate only locally and sum up local contributions to $-\Psi^T \frac{\partial R}{\partial s}$ (assembly).
- Local contributions:
  by hand, algorithmic differentiation, or even finite differences
  [S./Jimack 2008]

# Evaluation of required partial derivatives

- Non-trivial part is $-\Psi^T \frac{\partial R}{\partial s}$, $s$ are nodal coordinates.
- Jacobian $\frac{\partial R}{\partial s}$ is sparse.
  But needs not be build as a whole.
- Only required for one matrix-vector product $-\Psi^T \frac{\partial R}{\partial s}$.
  $\Rightarrow$ Cheaper to evaluate only locally and sum up local contributions to $-\Psi^T \frac{\partial R}{\partial s}$ (assembly).
- Local contributions:
  by hand, algorithmic differentiation, or even finite differences [S./Jimack 2008]

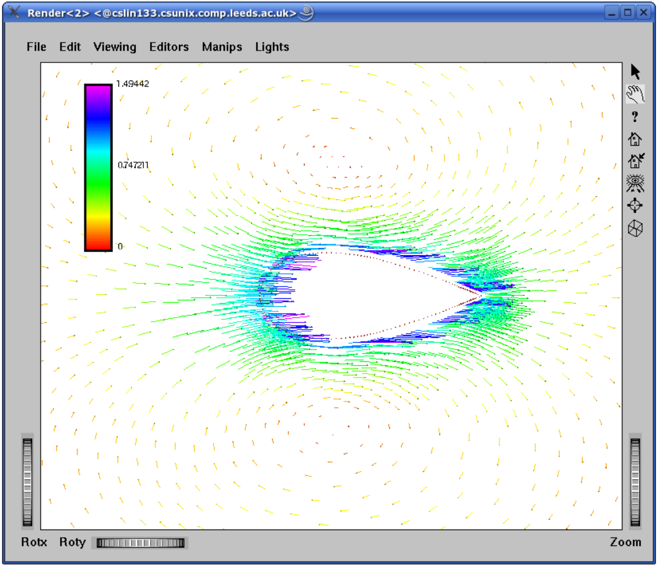
solution: velocities



# Example 3: Shape optimisation

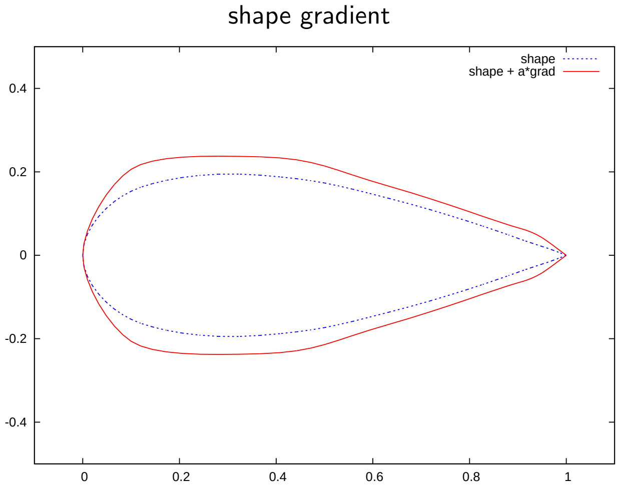
Navier Stokes, drag minimisation, $Re = 10$

adjoint: velocities



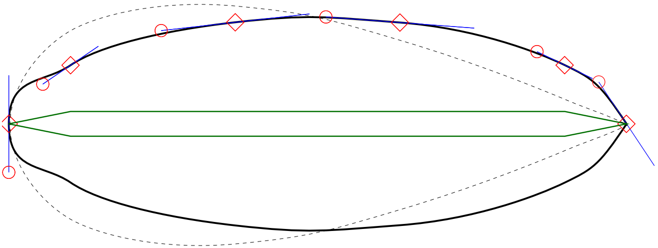
# Example 3: Shape optimisation

Navier Stokes, drag minimisation, $Re = 10$



# Example 3: Shape optimisation

Navier Stokes, drag minimisation, $Re = 10$

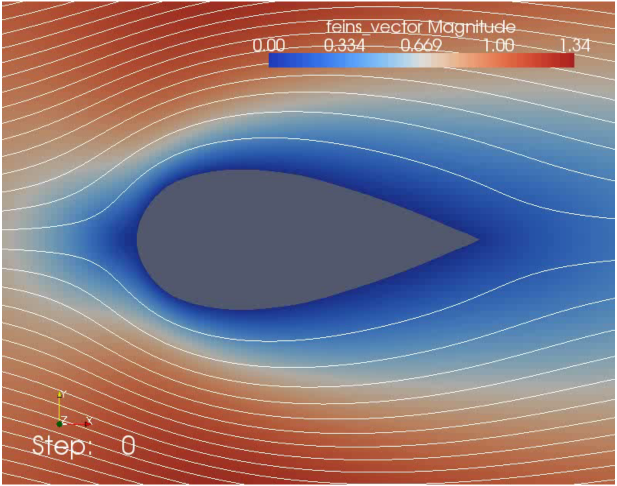



$f_0 = 1.4056 \qquad f_* = 1.3714 \qquad \Rightarrow 2.4\%$ reduced

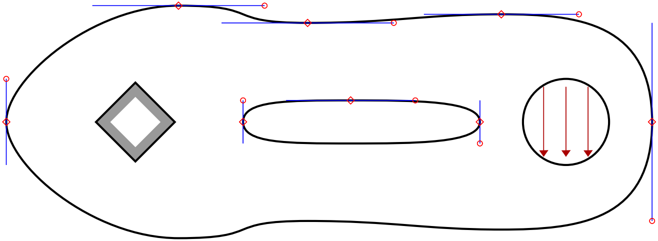
- 10 parameters for optimisation
- 18 SQP steps
- 69 function evaluations
- $\sim$ 145,000 DOFs in each nonlinear equation system,
  $J(s)$ : 9:20 min, adjoint 6:10 min

optimisation movie

$$f_0 = 1.4056 \qquad f_* = 1.3714$$



sketch of pedal crank problem:

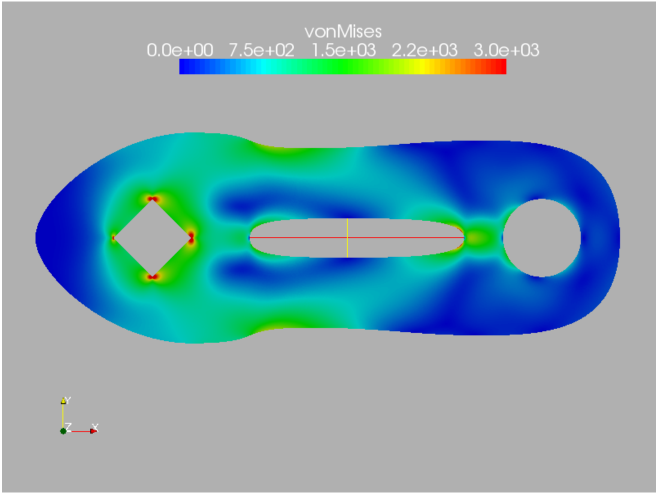


24 free parameters

performance functional: deformation energy

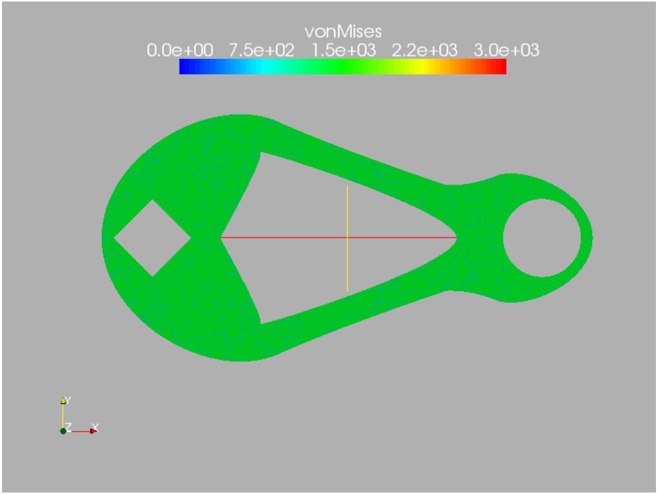
$$I(\mathcal{F}) := \frac{1}{2} a(u, u) - b(u) \quad + \alpha \int_{\Omega} 1 \, d\Omega$$

initial design



# Example 3: Shape optimisation
## Linear elasticity (with Andreas Günnel)

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



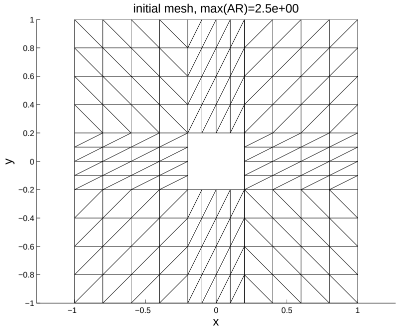
Singularly perturbed reaction diffusion problem

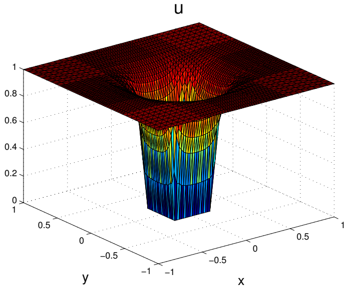
$$\begin{aligned} -\varepsilon^2 \Delta u + u &= 1 && \text{in } \Omega \\ u &= 0 && \text{on } \Gamma_D \\ \frac{\partial u}{\partial n} &= 0 && \text{on } \Gamma_N \end{aligned}$$

$$J := \sum_{T \in \mathcal{T}} J_{e,T}^2 \quad \text{local DWR error estimate}$$

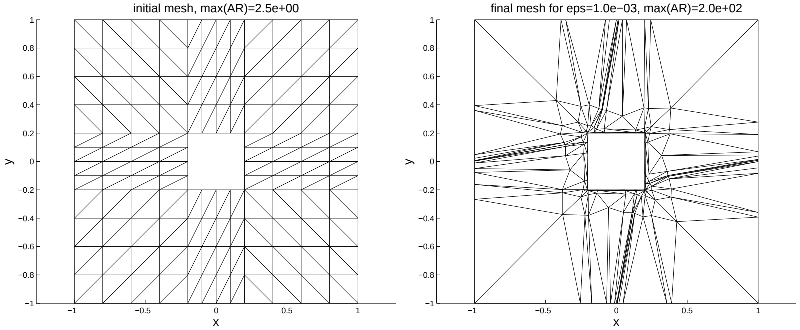
# Example 4: Mesh optimisation

Domain, Solution



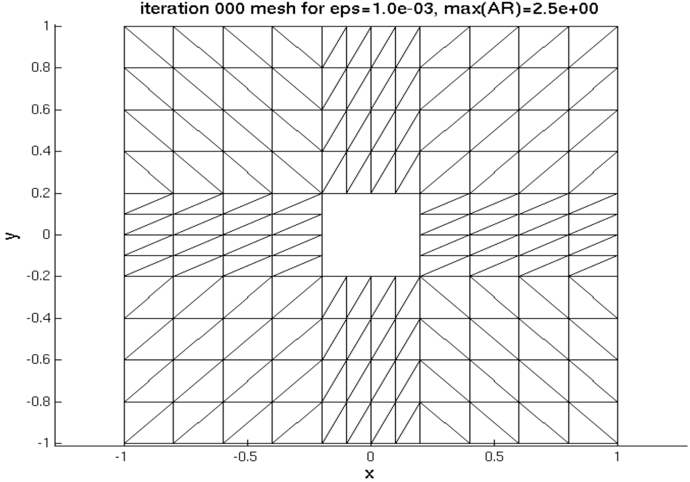



Meshes, $\varepsilon = 10^{-3}$
initial and optimised coarse mesh
256 DOFS for optimisation



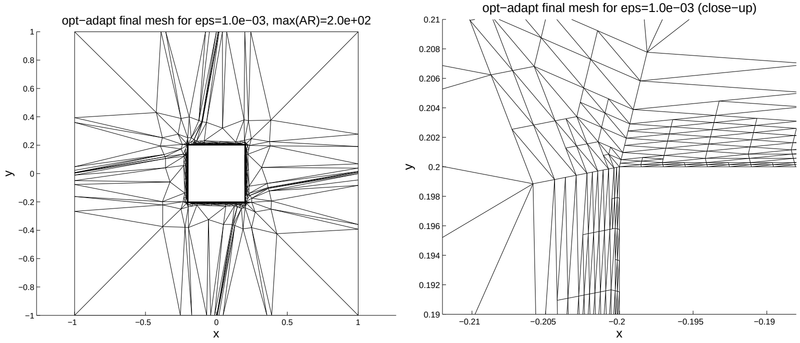
# Example 4: Mesh optimisation

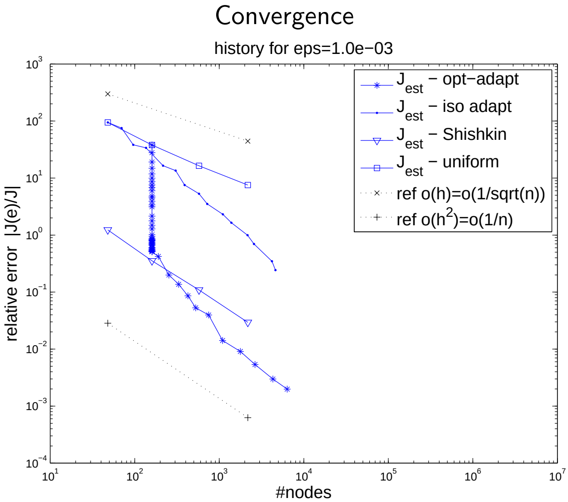
Meshes, $\varepsilon = 10^{-3}$



Meshes, $\varepsilon = 10^{-3}$

`opt-adapt`




Convergence
history for eps=1.0e−03
relative error |J(e)/J|
#nodes
J_est − opt−adapt
J_est − iso adapt
J_est − Shishkin
J_est − uniform
ref o(h)=o(1/sqrt(n))
ref o(h^2)=o(1/n)

# Conclusions

- Discrete-Adjoint-Technique gives a relatively simple way to extend an existing simulation code to include sensitivity analysis, and thus to allow optimisation.
  - Advantages: simple, reuse of code.
  - Disadvantages: optimisation of the discretised problem, rather than approximation of the optimal solution of the continuous problem.
- Technique very general, can be applied in very different settings [S., PhD Thesis 2006]

# Conclusions

- Discrete-Adjoint-Technique gives a relatively simple way to extend an existing simulation code to include sensitivity analysis, and thus to allow optimisation.
  - Advantages: simple, reuse of code.
  - Disadvantages: optimisation of the discretised problem, rather than approximation of the optimal solution of the continuous problem.
- Technique very general, can be applied in very different settings [S., PhD Thesis 2006]

# Conclusions

- Discrete-Adjoint-Technique gives a relatively simple way to extend an existing simulation code to include sensitivity analysis, and thus to allow optimisation.
  - Advantages: simple, reuse of code.
  - Disadvantages: optimisation of the discretised problem, rather than approximation of the optimal solution of the continuous problem.
- Technique very general, can be applied in very different settings [S., PhD Thesis 2006]

# Conclusions

- Discrete-Adjoint-Technique gives a relatively simple way to extend an existing simulation code to include sensitivity analysis, and thus to allow optimisation.
  - Advantages: simple, reuse of code.
  - Disadvantages: optimisation of the discretised problem, rather than approximation of the optimal solution of the continuous problem.
- Technique very general, can be applied in very different settings [S., PhD Thesis 2006]

Thank you!

Congratulations Peter!

# Hope you to see you in Chemnitz now and then.