UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Memory-reduced Adjoint Calculation

Andrea Walther
Institut für Mathematik
Universität Paderborn

Woudschoten Conference 2012

October 3–5, 2012

Outline

Starting Point

For nonlinear problems:

Adjoints are great

because of the time complexity to compute gradients for

- $\min f(x)$
- $\widehat{J}'(u, x(u))$ such that

  $$x'(u) = f(x(u), u) \; + \; \text{IC}$$

  using the adjoint equation

Constrained Optimization Problem

- Task: $\min_x f(x), \quad \text{s.t.} \quad c(x) = 0$
- Solution: Consider Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$ and compute

$$0 = \nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k) = \left[\nabla f(x) + \lambda^T A(x),\ c(x)\right] \in \mathbb{R}^{n+m},$$

with $A(x) = \nabla c(x)$

Constrained Optimization Problem

- Task: $\min_x f(x), \quad \text{s.t.} \quad c(x) = 0$
- Solution: Consider Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$ and compute

$$0 = \nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k) = \left[\nabla f(x) + \lambda^T A(x),\ c(x)\right] \in \mathbb{R}^{n+m},$$

with $A(x) = \nabla c(x)$, e.g., using an SQP method $\Rightarrow$

$$\nabla^2_{x,\lambda} \mathcal{L}(x_k, \lambda_k)\, p_k^N = \begin{bmatrix} B(x_k, \lambda_k) & A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} p_k^N = -\nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k)$$

Constrained Optimization Problem

- Task: $\min_x f(x), \quad \text{s.t.} \quad c(x) = 0$
- Solution: Consider Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$ and compute

$$0 = \nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k) = \left[\nabla f(x) + \lambda^T A(x),\ c(x)\right] \in \mathbb{R}^{n+m},$$

with $A(x) = \nabla c(x)$, e.g., using an SQP method $\Rightarrow$

$$\nabla^2_{x,\lambda} \mathcal{L}(x_k, \lambda_k)\, p_k^N = \begin{bmatrix} B(x_k, \lambda_k) & A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} p_k^N = -\nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k)$$

- Iterative solve requires $A(x_k)v$, $A(x_k)^\top w$, and $\lambda^\top A(x_k)$!!

Starting Point: Adjoints are Great!

BUT

$$\text{MEM}(\bar{y}^\top F'(x)) \quad \sim \quad \text{OPS}(F(x))$$

independent of chosen approach!!

Starting Point: Adjoints are Great!

BUT

$$\text{MEM}(\bar{y}^\top F'(x)) \quad \sim \quad \text{OPS}(F(x))$$

independent of chosen approach!!

- For large-scale problems not acceptable!!
- Even for smaller problems long runtime due to extensive memory access.

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations
- $F(x)$ = Newton-like solve $\Rightarrow$ [Griewank, Walther 2008]

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations
- $F(x)$ = Newton-like solve $\Rightarrow$ [Griewank, Walther 2008]
- $F(x)$ = Krylov subspace method ???

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations
- $F(x)$ = Newton-like solve $\Rightarrow$ [Griewank, Walther 2008]
- $F(x)$ = Krylov subspace method ???

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
    $\Rightarrow$ checkpointing
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations
- $F(x)$ = Newton-like solve $\Rightarrow$ [Griewank, Walther 2008]
- $F(x)$ = Krylov subspace method ???

Alternatives:

- no assumptions on structure of $F(x)$
  For AD tools:
  subroutine-oriented checkpointing (Naumann, Utke, ...)
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
    $\Rightarrow$ checkpointing
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations
    $\Rightarrow$ adapted derivative calculation
- $F(x)$ = Newton-like solve $\Rightarrow$ [Griewank, Walther 2008]
- $F(x)$ = Krylov subspace method ???

Gradient Calculation (Time Integration)

1. Forward integration:

   $x_{i+1} = F_i(x_i, u_i),\ i = 1, \ldots, l$

   with $x_i \in \mathbb{R}^n$ state, u_i control

   number of time steps l might be not known a priori

2. Evaluation of target function $J(x, u) \to \widehat{J}(u)$

3. Backward integration:

   $\bar{x}_{i-1} = \bar{F}_i(\bar{x}_i, x_{i-1}, u_{i-1}, u_i),\ i = l, \ldots, 1$

   AD, discretization of continuous adjoint, hand coded, ...

4. Gradient calculation

Gradient Calculation (Time Integration)

1. Forward integration:

   $x_{i+1} = F_i(x_i, u_i),\ i = 1, \ldots, l$

   with $x_i \in \mathbb{R}^n$ state, u_i control

   number of time steps l might be not known a priori

2. Evaluation of target function $J(x, u) \to \widehat{J}(u)$

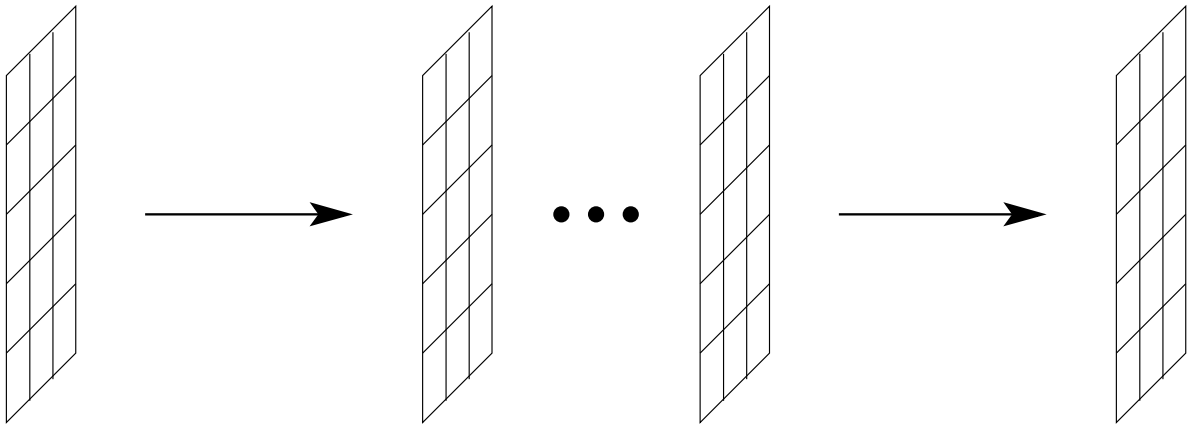
3. Backward integration:

   $\bar{x}_{i-1} = \bar{F}_i(\bar{x}_i, x_{i-1}, u_{i-1}, u_i),\ i = l, \ldots, 1$

   AD, discretization of continuous adjoint, hand coded, ...

4. Gradient calculation

Calculating Adjoints

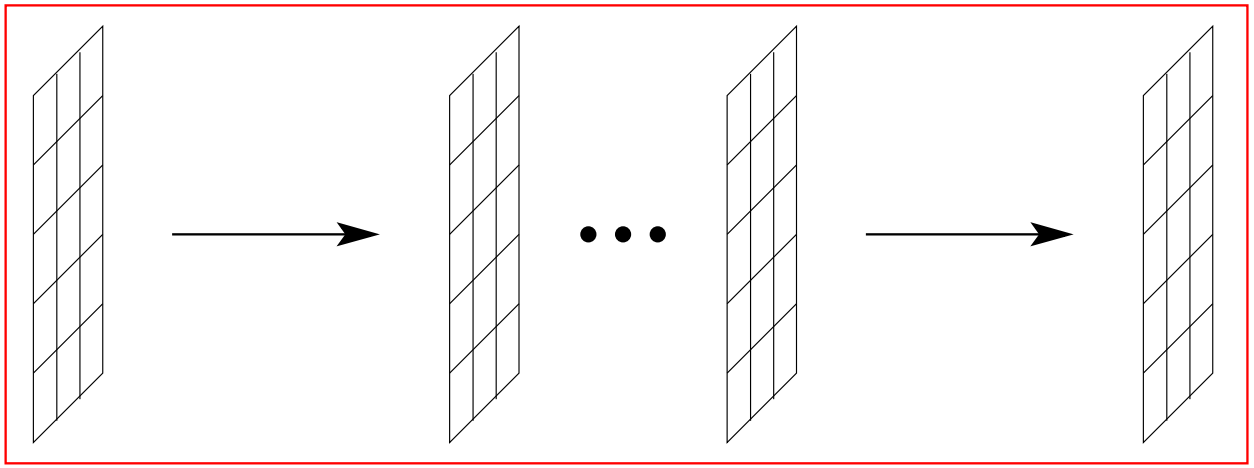


Integration of forward solution:

$$x_{i+1} = F_i(x_i, u_i), \qquad i = 1, \ldots, l$$

Integration of adjoint $\bar{x}_{i-1} = \bar{F}_i(\bar{x}_i, \bar{u}_i, x_{i-1}),\ i = l, \ldots, 1$?

Calculating Adjoints



Integration of forward solution:

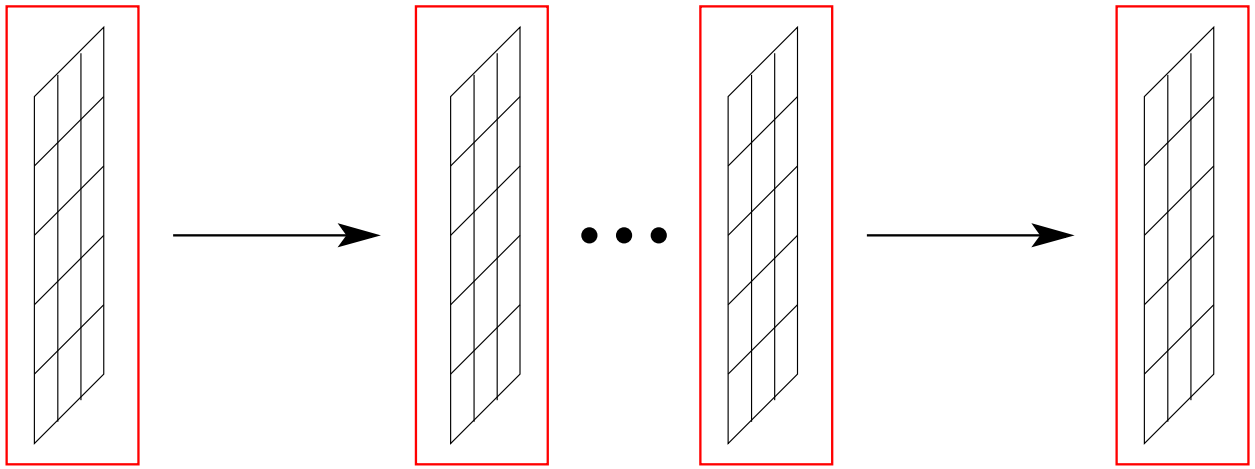
$$x_{i+1} = F_i(x_i, u_i), \qquad i = 1, \ldots, l$$

Integration of adjoint $\bar{x}_{i-1} = \bar{F}_i(\bar{x}_i, \bar{u}_i, x_{i-1}),\ i = l, \ldots, 1$?

"Black-Box"-approach, e.g. using AD

Memory requirement?? Computing time ??

Calculating Adjoints



Integration of forward solution:

$$x_{i+1} = F_i(x_i, u_i), \qquad i = 1, \ldots, l$$

Integration of adjoint $\bar{x}_{i-1} = \bar{F}_i(\bar{x}_i, \bar{u}_i, x_{i-1}),\ i = l, \ldots, 1$?

Time Structure Exploitation

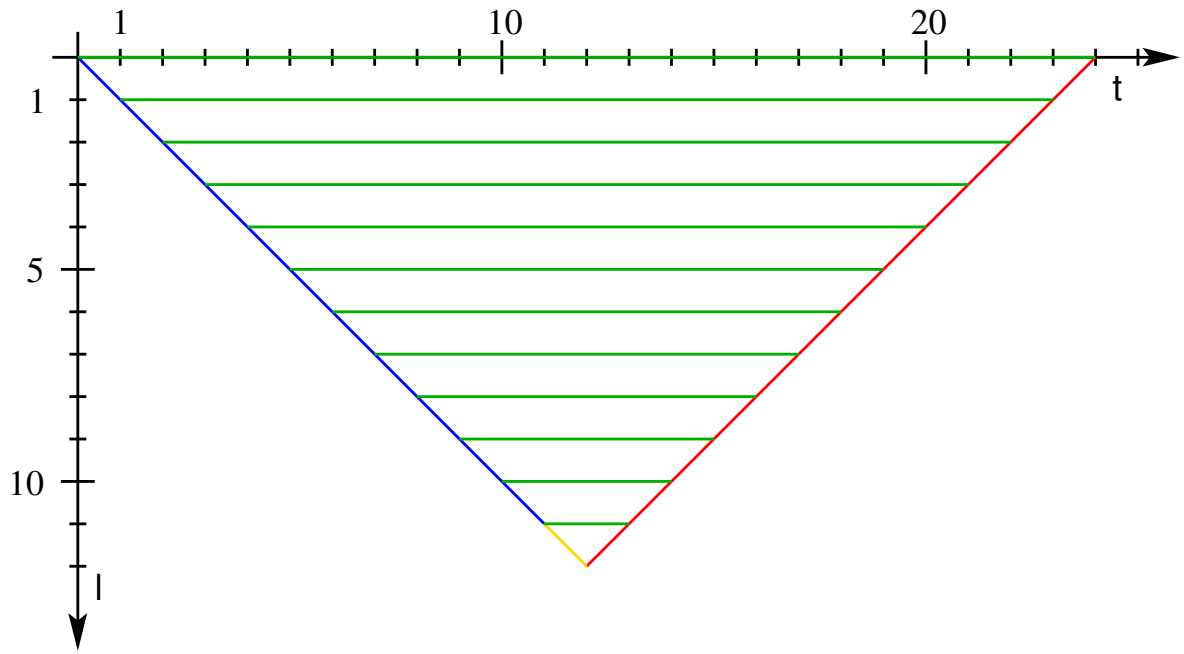
Memory requirement?? Computing time ?? Adjoint ??

Store-Everything Approach

Example: 12 time steps

Store-Everything Approach

Example: 12 time steps



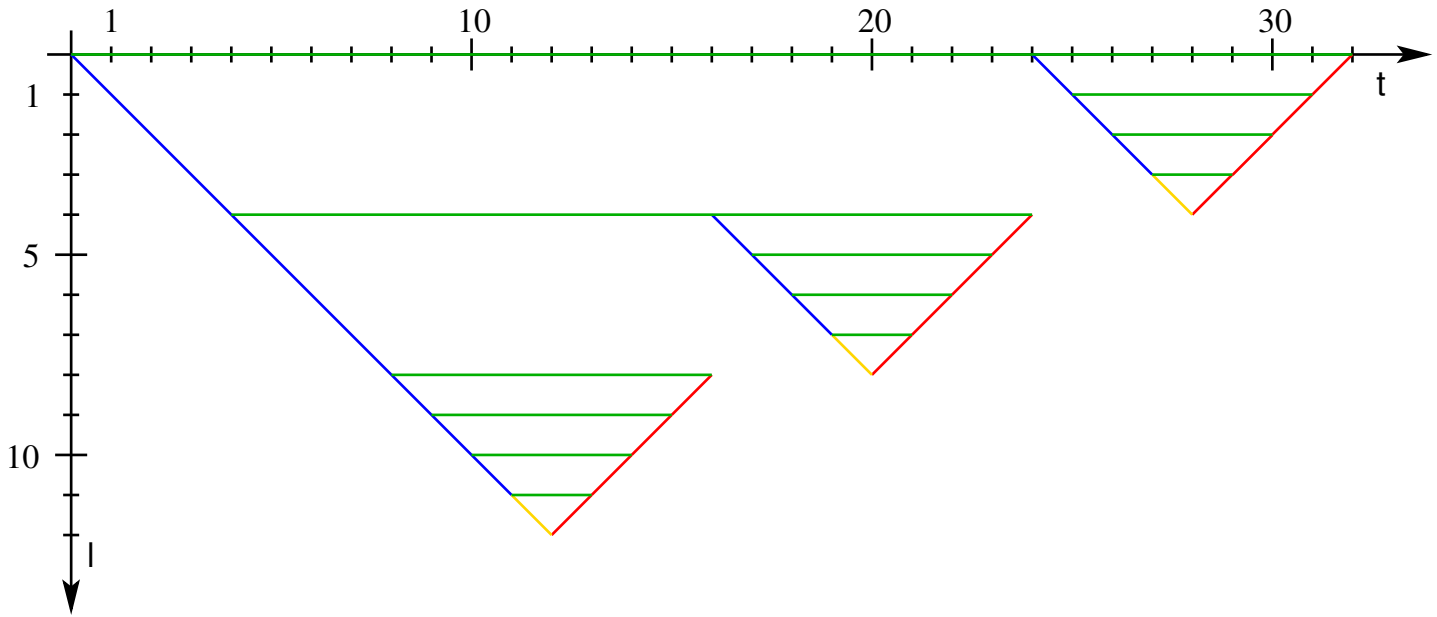
→ MEM = $O(l)$, TIME = 2l,
might cause problems, even if it fits in memory

Windowing / Multilevel-Checkpointing

Example: 12 time steps

Windowing / Multilevel-Checkpointing

Example: 12 time steps



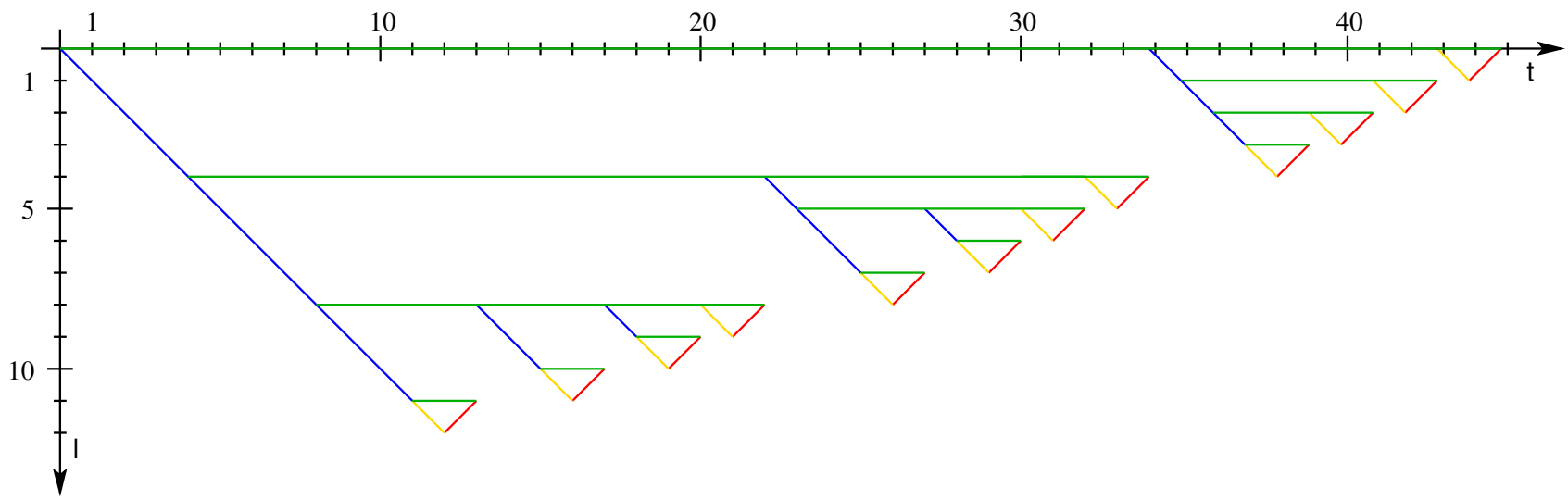
→ MEM = c, TIME $\approx$ 3l, here: c = 6, additional #eval(F_i) = 17

Binomial Checkpointing

Example: 12 time steps, 4 checkpoints, reusage of all checkpoints!

Binomial Checkpointing

Example: 12 time steps, 4 checkpoints, reusage of all checkpoints!



Provable optimal strategy
[Griewank, Walther '00, Walther, Griewank '04]

Theorem (Complexity Result for Binomial Checkpointing)

Given

$$\begin{aligned} l &= \text{total number of time steps to be reversed} \\ c &= \text{number of checkpoints that can be used} \\ r &= \text{unique integer satisfying} \end{aligned}$$

$$\beta(c, r-1) \;<\; l \;\leq\; \beta(c, r) \;\equiv\; \binom{c+r}{c}$$

Then: The minimal number of time step evaluations required equals

$$t(c, l) \;=\; rl - \beta(c+1, r-1) + 1$$

Proof: Based on induction [Griewank, Walther 2000]

Theorem (Complexity Result for Binomial Checkpointing)

Given

$$\begin{aligned} l &= \text{total number of time steps to be reversed} \\ c &= \text{number of checkpoints that can be used} \\ r &= \text{unique integer satisfying} \end{aligned}$$

$$\beta(c, r-1) \;<\; l \;\leq\; \beta(c, r) \;\equiv\; \binom{c+r}{c}$$

Then: The minimal number of time step evaluations required equals

$$t(c, l) \;=\; rl - \beta(c+1, r-1) + 1$$

Proof: Based on induction [Griewank, Walther 2000]

Implementation: revolve

Test Case: Optimal Control

Task:
Minimize cost function $\tilde{J}(u) := J(y(u), u)$ for

$$\begin{aligned} y_t - \frac{1}{\mathrm{Re}}\Delta y + (y \cdot \nabla)y + \nabla p &= Bu \\ -\mathrm{div}\ y &= 0 \text{ in } \Omega \times (0,T) \\ y(x,t) = 0 \text{ on } \partial\Omega \times (0,T) \quad y(x,0) &= y_0 \text{ in } \partial\Omega. \end{aligned}$$

Test Case: Optimal Control

Task:
Minimize cost function $\tilde{J}(u) := J(y(u), u)$ for

$$\begin{aligned} y_t - \frac{1}{\mathrm{Re}}\Delta y + (y \cdot \nabla)y + \nabla p &= Bu \\ -\mathrm{div}\ y &= 0 \text{ in } \Omega \times (0,T) \\ y(x,t) = 0 \text{ on } \partial\Omega \times (0,T) \quad y(x,0) &= y_0 \text{ in } \partial\Omega. \end{aligned}$$

$\Rightarrow$ Gradient $\hat{J}'(u) = J_u(y(u), u) - B^*\lambda$ required with

$$\begin{aligned} -\lambda_t - \frac{1}{\mathrm{Re}}\Delta\lambda - (y \cdot \nabla)\lambda + (\nabla y)^t\lambda + \nabla\xi &= -J_y^{(t)}(y(u), u) \\ -\mathrm{div}\ \lambda &= 0 \text{ in } \Omega \times (0,T) \\ \lambda(x,t) = 0 \text{ on } \partial\Omega \times (0,T) \quad \lambda(x,T) &= -J_y^{(T)}(y(u), u) \text{ in } \partial\Omega. \end{aligned}$$

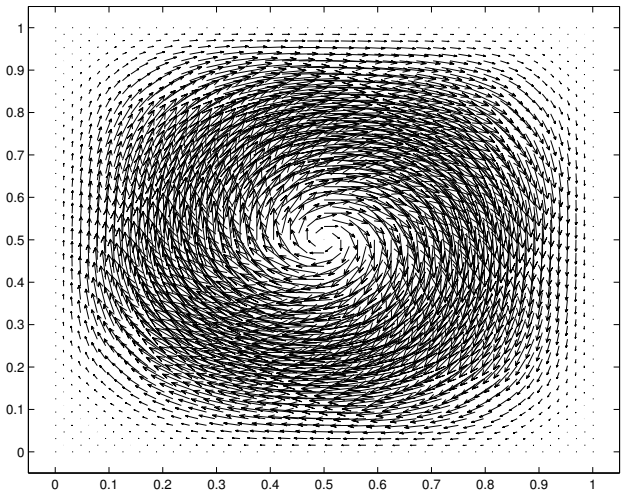
Test Case: Cavity Flow

joint work with Michael Hinze

Cost function of tracking typ, initial state:

$$y_0(x) = e \begin{bmatrix} (\cos 2\pi x_1 - 1)\sin 2\pi x_2 \\ -(\cos 2\pi x_2 - 1)\sin 2\pi x_1 \end{bmatrix},$$

$\Omega := (0,1) \times (0,1)$, $T = 1$, Re = 10, e = Euler number



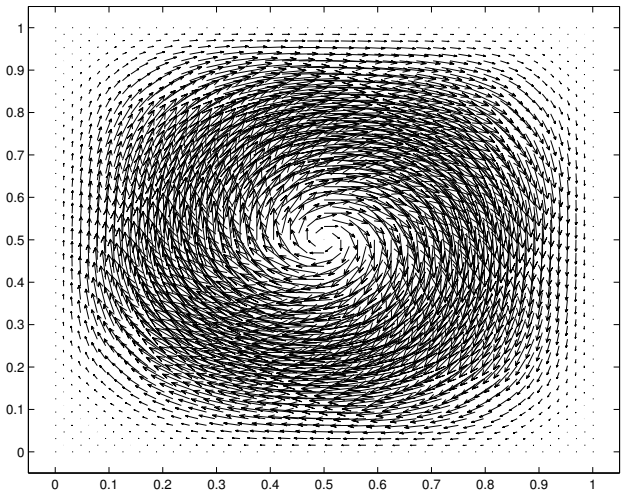
Test Case: Cavity Flow

joint work with Michael Hinze

Cost function of tracking typ, initial state:

$$y_0(x) = e \begin{bmatrix} (\cos 2\pi x_1 - 1)\sin 2\pi x_2 \\ -(\cos 2\pi x_2 - 1)\sin 2\pi x_1 \end{bmatrix},$$

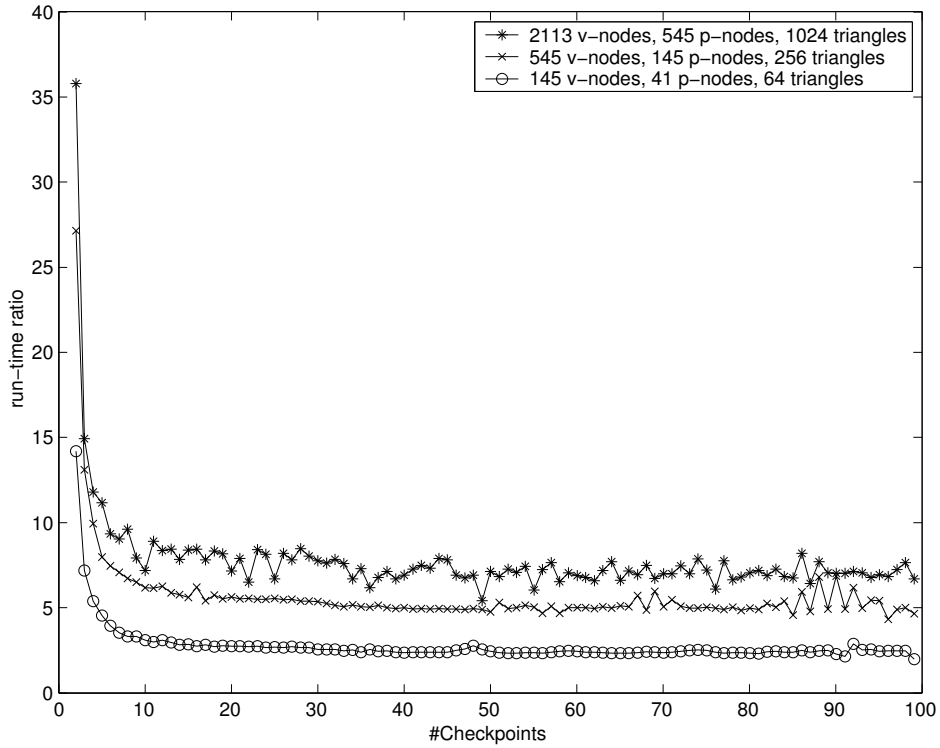
$\Omega := (0,1) \times (0,1)$, $T = 1$, Re = 10, e = Euler number



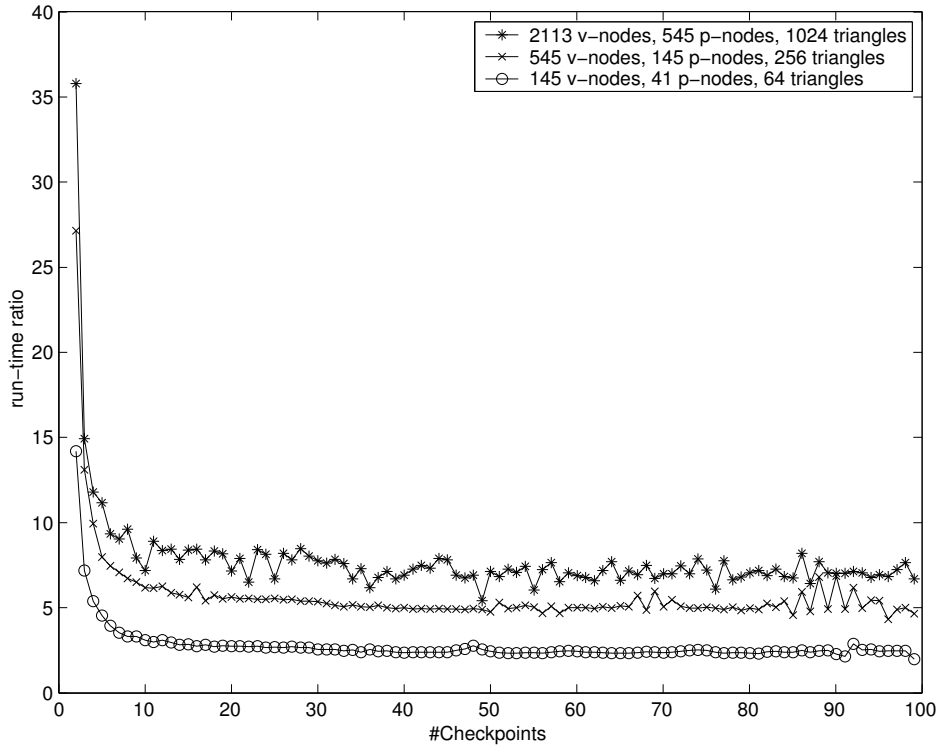
Desired state:

$$\begin{aligned} z(x,t) &= \begin{bmatrix} \varphi_{x_2}(x,t) \\ -\varphi_{x_1}(x,t) \end{bmatrix}, \\ \varphi(x,t) &= \theta(x_1,t)\theta(x_2,t), \\ \theta(y,t) &= (1-y)^2(1-\cos kt), \\ y &\in [0,1]. \end{aligned}$$

Runtime Ratios, 1000 Time Steps



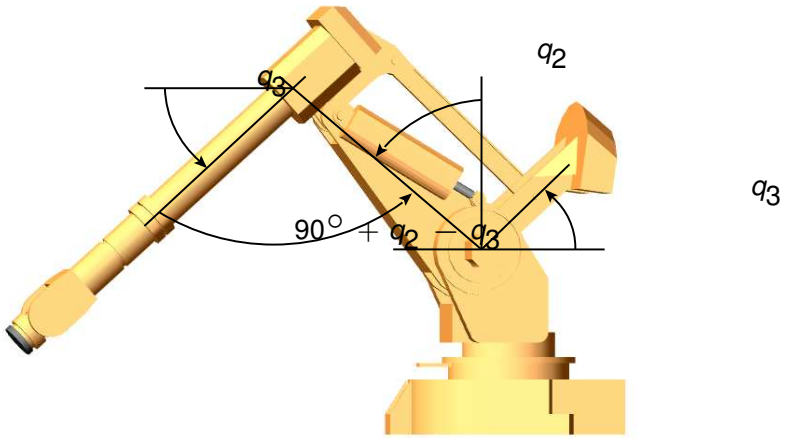
Runtime Ratios, 1000 Time Steps



- Run-time ratio between 2 and 7.
- Only a very small number of checkpoints required!

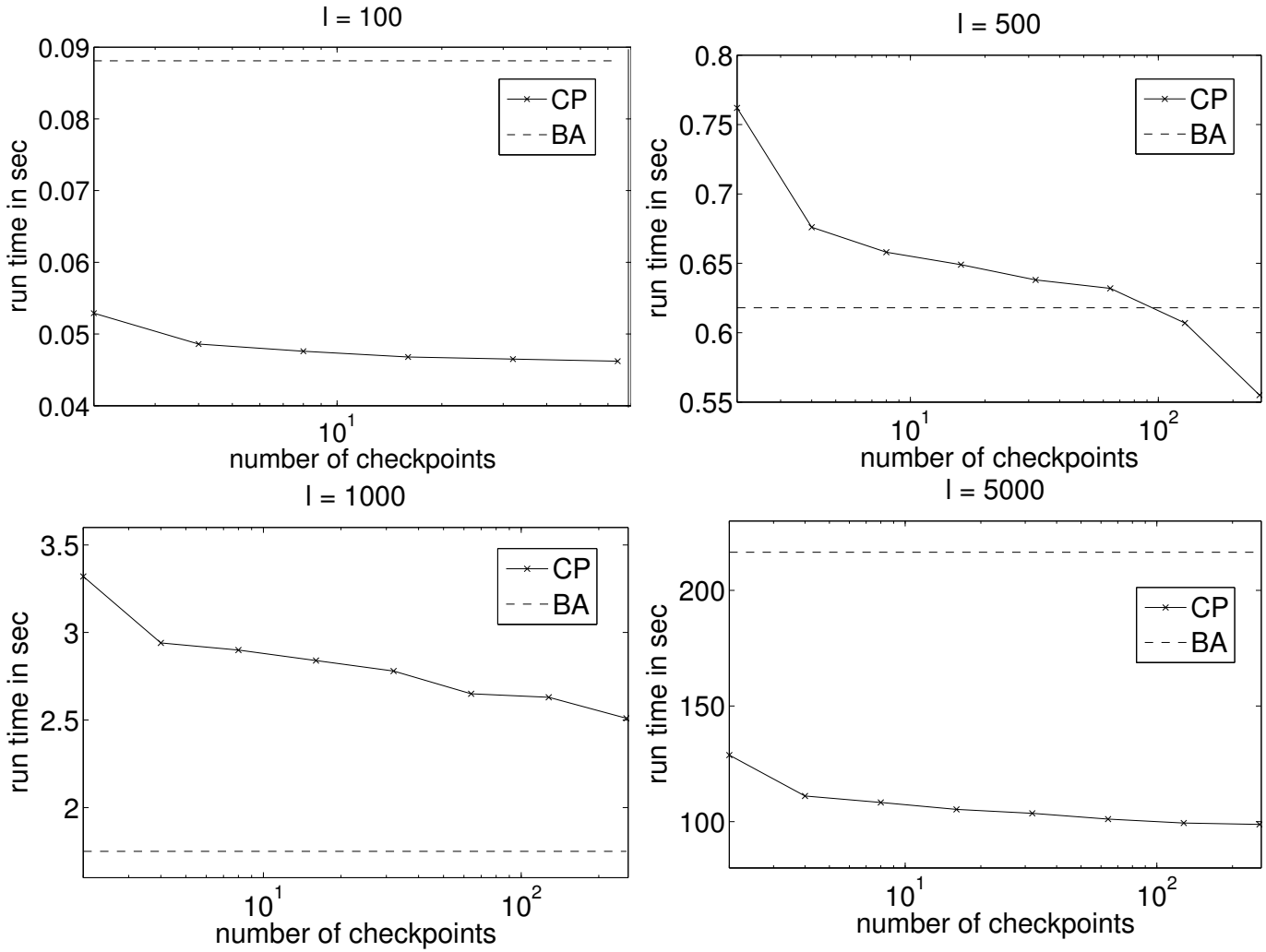
Optimal turn-around maneuver of an industrial robot

- Industrial robot ABB IRB 6400 (Büskens et al.)
- $q = (q_1, q_2, q_3)$ as angular coordinates of the robot's joints
- Controlled via three functions u_1 through u_3
- Control problem: minimize the energy-related objective



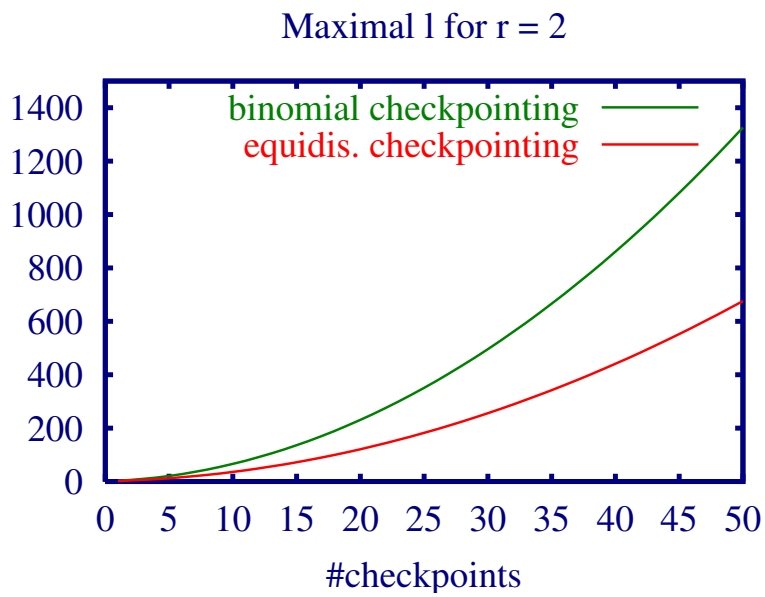
$$J(q,u) = \int_0^{t_f} [u_1(t)^2 + u_2(t)^2 + u_3(t)^2]\, dt$$

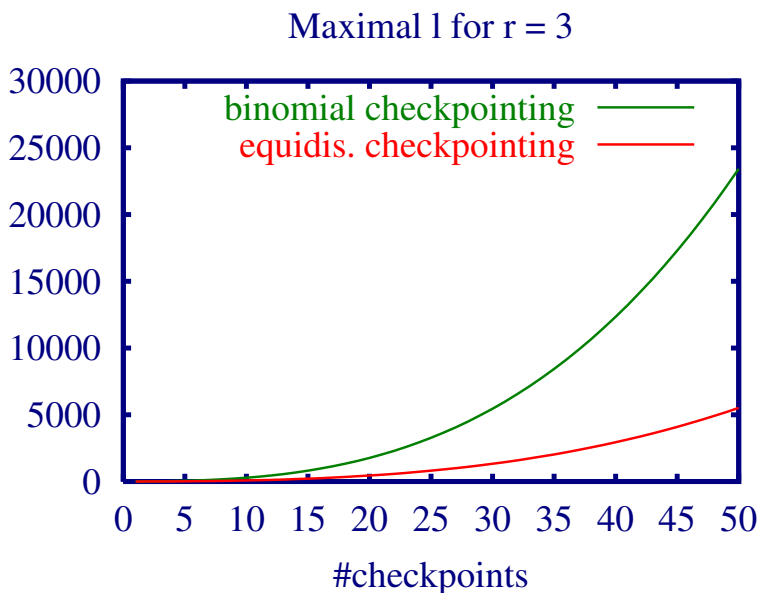
- Trajectory approximation using the standard Runge-Kutta method of order 4 resulting in $\approx$ 800 lines of C++ code
- Adjoint computation: Combination of ADOL-C and revolve



[Kowarz, Walther 2007]

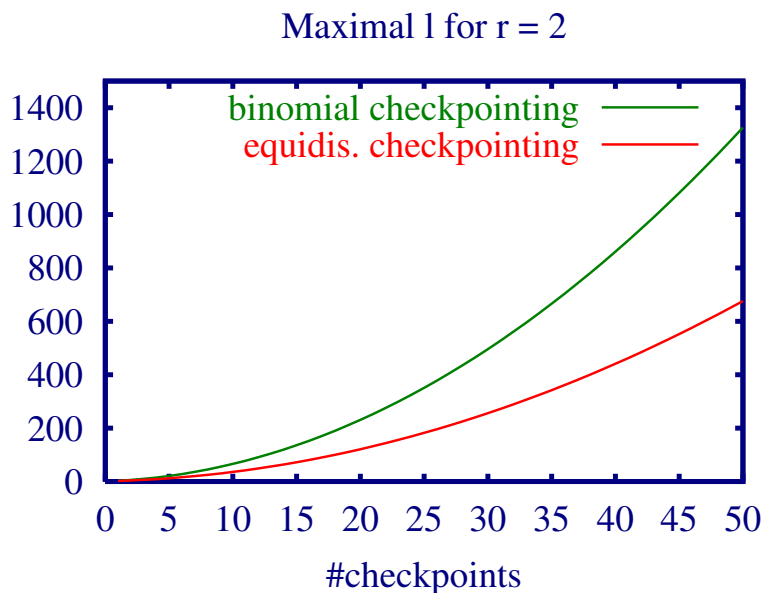
Repetition Number *r* vs Time Step Number *l*

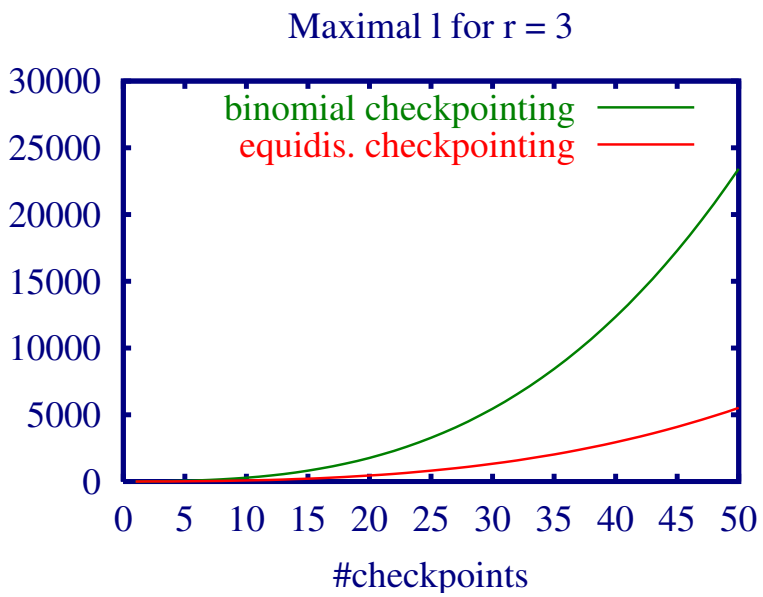




[Walther, Griewank 2004]

Repetition Number *r* vs Time Step Number *l*

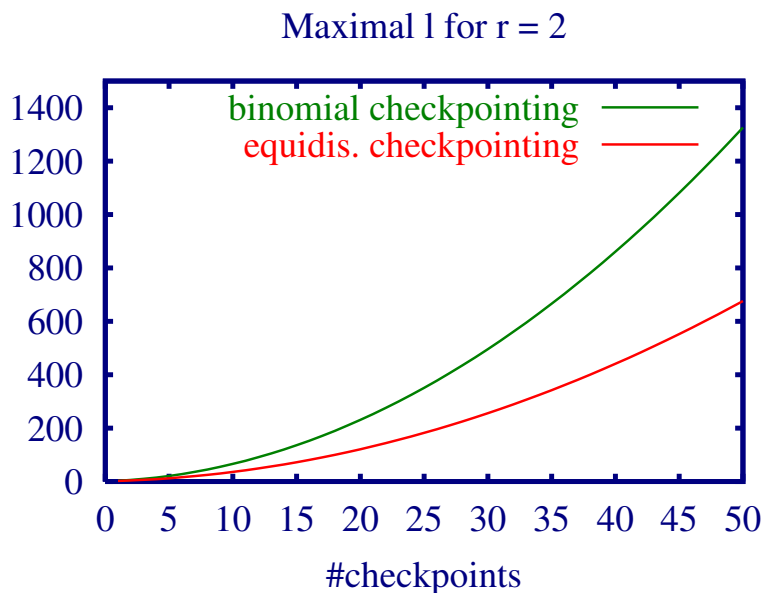


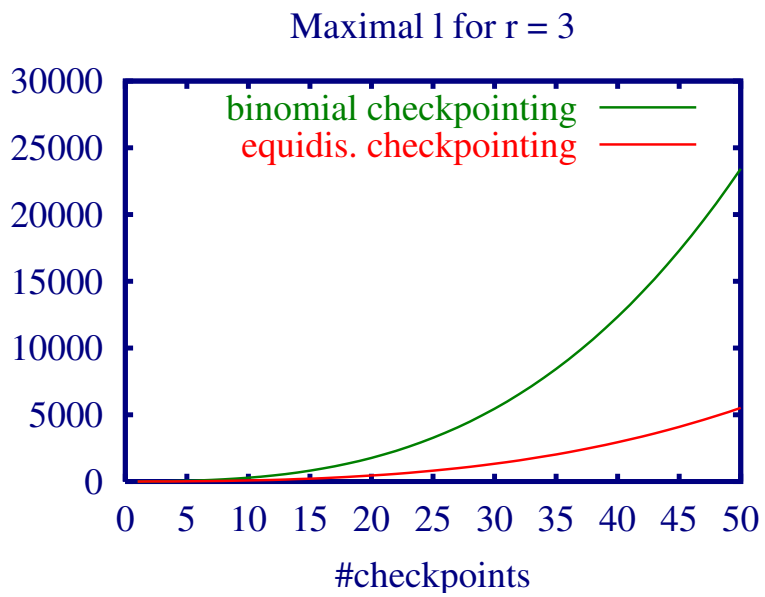


[Walther, Griewank 2004]

Problem: Extension for unknown *l*?

Repetition Number *r* vs Time Step Number *l*





[Walther, Griewank 2004]

Problem: Extension for unknown *l*?

- **arevolve** (Sternberg, Griewank, Walther), based on heuristics
- Optimal strategies? Analyse schedules for $r < 4$!

Theorem (Complexity Result for Online Checkpointing I)

Given

$$\begin{aligned} c = &\quad \textit{number of checkpoints that can be used} \\ l = &\quad \textit{unknown number of time steps to be reversed} \\ &\quad \textit{with } l \leq \beta(c,2) \equiv u_2 \end{aligned}$$

Then: The minimal number of time step evaluations required by the online checkpointing equals

$$t(l,c) \;=\; rl - \beta(c+1, r-1) \qquad r \in \{1,2\}.$$

Proof: Based on induction:
Easy for $c = 1$
$c-1 \Rightarrow c$: Verify assumption for $l \leq 2c+1$, then use result for $c-1$

[Stumm, Walther 2010]

Theorem (Complexity Result for Online Checkpointing II)

Given

$c =$ *number of checkpoints that can be used*
$l =$ *unknown number of time steps to be reversed with* $l \leq \beta(c,3) \equiv u_3$

Then: The number of time step evaluations required by the online checkpointing equals

$$\tilde{t}(l,c) \leq t(l,c) + 1$$

Proof: Based on induction using offline checkpointing for $l = \beta(c,3)$ [Stumm, Walther 2010]

Resulting Algorithm for $r = 2$

Resulting Algorithm for $r = 3$

Summary: Theoretical Results

Provable (almost) optimal online checkpointing strategies for

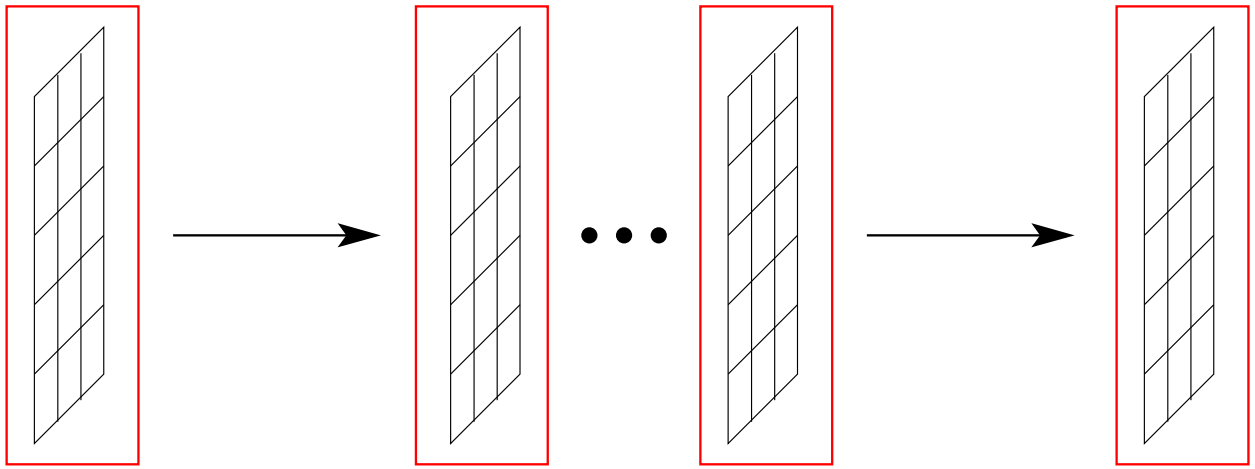
c	10	20	40	80	160	320
u_2	66	231	861	3321	13041	51681
u_3	286	1771	12341	91881	708561	5564321

For larger values of *l*: [Moin, Wang 2010]

Implementation: Combination of

- Adjoints of time integration and
- routine **revolve** (C/Fortran version)

Calculating Adjoints II



Integration of forward solution:

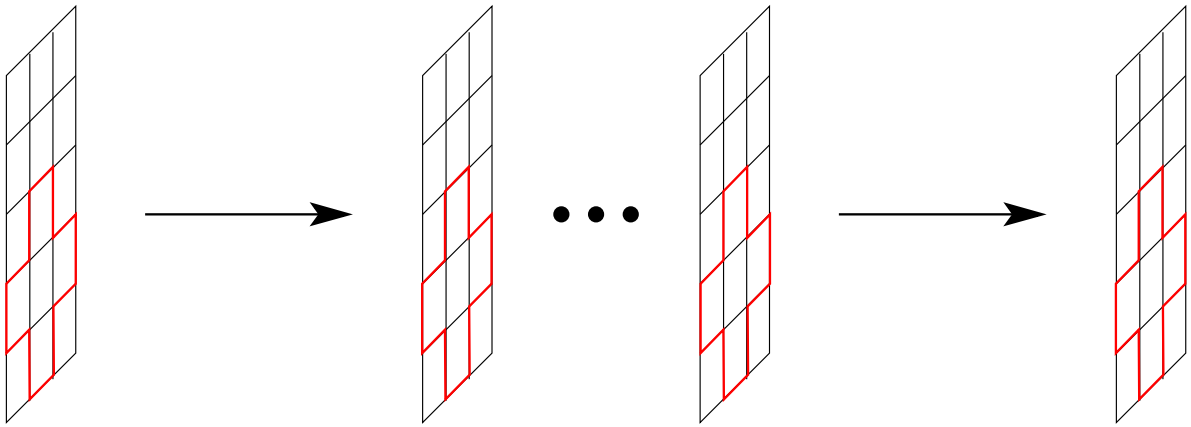
$$y_{i+1} = F_i(y_i, u_i), \qquad i = 1, \ldots, l$$

Integration of adjoint $\bar{y}_{i-1} = \bar{F}_i(\bar{y}_i, \bar{u}_i, y_i),\ i = l, \ldots, 1$?

Time Structure Exploitation

Memory requirement?? Computing time ?? Adjoint ??

Calculating Adjoints II



Integration of forward solution:

$$y_{i+1} = F_i(y_i, u_i), \qquad i = 1, \ldots, l$$

Integration of adjoint $\bar{y}_{i-1} = \bar{F}_i(\bar{y}_i, \bar{u}_i, y_i),\ i = l, \ldots, 1$?

Time and Space Structure Exploitation

Memory requirement?? Computing time ?? Adjoint ??

Nanooptics: Optimisation

So far: Genetic algorithms

Now: L-BFGS and efficient gradient computation

- ADOL-C coupled with hand-coded adjoints
- Checkpointing (160 000 time steps!!)

$\Rightarrow$ TIME(gradient)/TIME(target function) $<$ 7 despite of checkpointing!

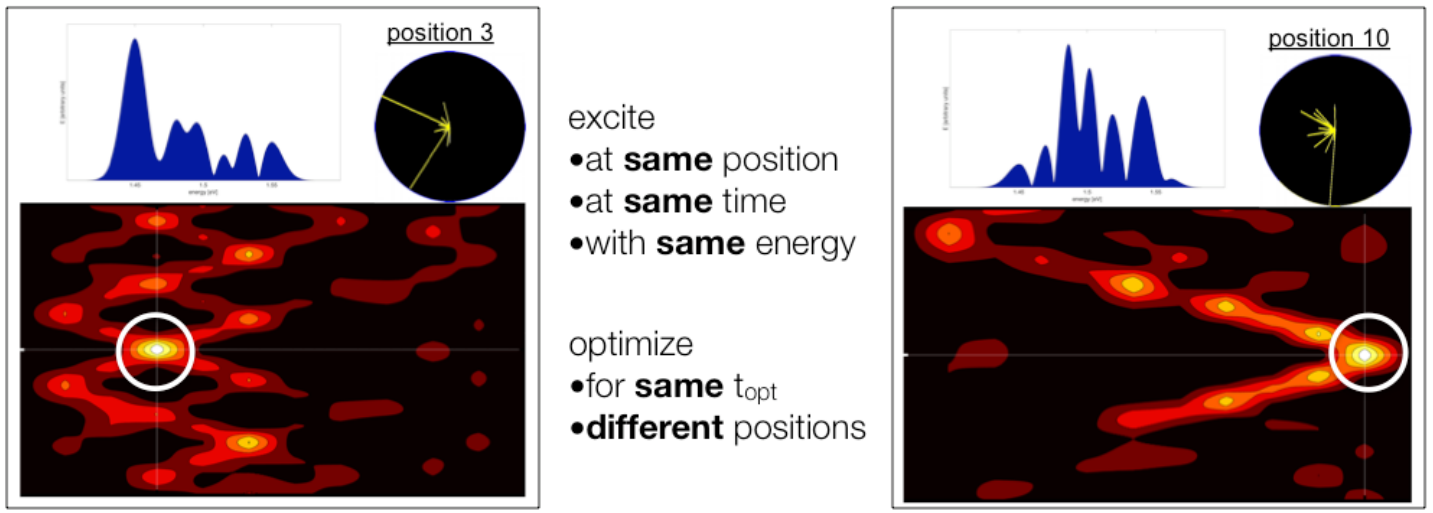
Nanooptics: Optimisation

So far: Genetic algorithms

Now: L-BFGS and efficient gradient computation

- ADOL-C coupled with hand-coded adjoints
- Checkpointing (160 000 time steps!!)

$\Rightarrow$ TIME(gradient)/TIME(target function) $<$ 7 despite of checkpointing!

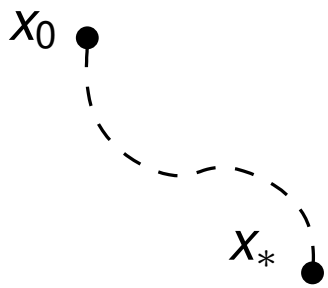


Adjoints for Pseudo-Timestepping

Time-stepping prozess =
fixed point iteration for state x at given control u

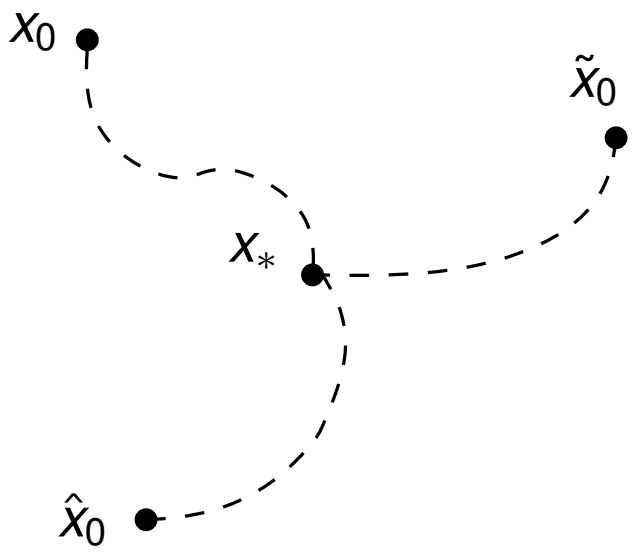
Adjoints for Pseudo-Timestepping

Time-stepping prozess =
fixed point iteration for state x at given control u



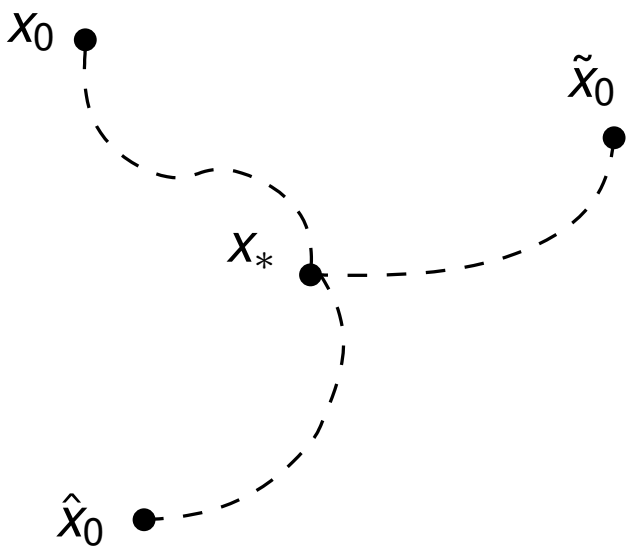
Adjoints for Pseudo-Timestepping

Time-stepping prozess =
fixed point iteration for state x at given control u



Adjoints for Pseudo-Timestepping

Time-stepping prozess =
fixed point iteration for state x at given control u



No need to store intermediate results!!
Derivative calculation??

Calculation of Adjoints for Fixpoint Iterations

Fixpoint equation $x_* = F(x_*, u)$

Desired derivatives:

$$\frac{\partial x_*}{\partial u} = \frac{\partial F(x_*, u)}{\partial u}$$

Calculation of Adjoints for Fixpoint Iterations

Fixpoint equation $x_* = F(x_*, u)$

Desired derivatives:

$$\frac{\partial x_*}{\partial u} = \frac{\partial F(x_*, u)}{\partial u}$$

This yields:

$$\frac{\partial x_*}{\partial u} = \frac{\partial F(x_*, u)}{\partial x}\frac{\partial x_*}{\partial u} + \frac{\partial F(x_*, u)}{\partial u}$$

$$\frac{\partial x_*}{\partial u} = \left(I - \frac{\partial F(x_*, u)}{\partial x}\right)^{-1} \frac{\partial F(x_*, u)}{\partial u}$$

Calculation of Adjoints for Fixpoint Iterations

Fixpoint equation $x_* = F(x_*, u)$

Desired derivatives:

$$\frac{\partial x_*}{\partial u} = \frac{\partial F(x_*, u)}{\partial u}$$

This yields:

$$\begin{aligned}
\frac{\partial x_*}{\partial u} &= \frac{\partial F(x_*, u)}{\partial x}\frac{\partial x_*}{\partial u} + \frac{\partial F(x_*, u)}{\partial u} \\
\frac{\partial x_*}{\partial u} &= \left(I - \frac{\partial F(x_*, u)}{\partial x}\right)^{-1} \frac{\partial F(x_*, u)}{\partial u} \\
\bar{x}^T \frac{\partial x_*}{\partial u} &= \underbrace{\bar{x}^T \left(I - \frac{\partial F(x_*, u)}{\partial x}\right)^{-1}}_{=: \xi_*^T} \frac{\partial F(x_*, u)}{\partial u}
\end{aligned}$$

Fixpoint Iteration for Adjoints

With

$$\xi_*^T = \bar{x}^T \left(I - \frac{\partial F(x_*, u)}{\partial x} \right)^{-1}$$

one obtains

$$\xi_*^T \left(I - \frac{\partial F(x_*, u)}{\partial x} \right) = \bar{x}^T \quad \Leftrightarrow \quad \xi_*^T = \xi_*^T \frac{\partial F(x_*, u)}{\partial x} + \bar{x}^T \quad .$$

Fixpoint Iteration for Adjoints

With

$$\xi_*^T = \bar{x}^T \left(I - \frac{\partial F(x_*, u)}{\partial x} \right)^{-1}$$

one obtains

$$\xi_*^T \left(I - \frac{\partial F(x_*, u)}{\partial x} \right) = \bar{x}^T \quad \Leftrightarrow \quad \xi_*^T = \xi_*^T \frac{\partial F(x_*, u)}{\partial x} + \bar{x}^T \quad .$$

→ Fixpoint iteration for ξ_*^T:

$$\xi_{k+1}^T = \xi_k^T \frac{\partial F(x_*, u)}{\partial x} + \bar{x}^T$$

Convergence

Theorem (Christianson '94)

For

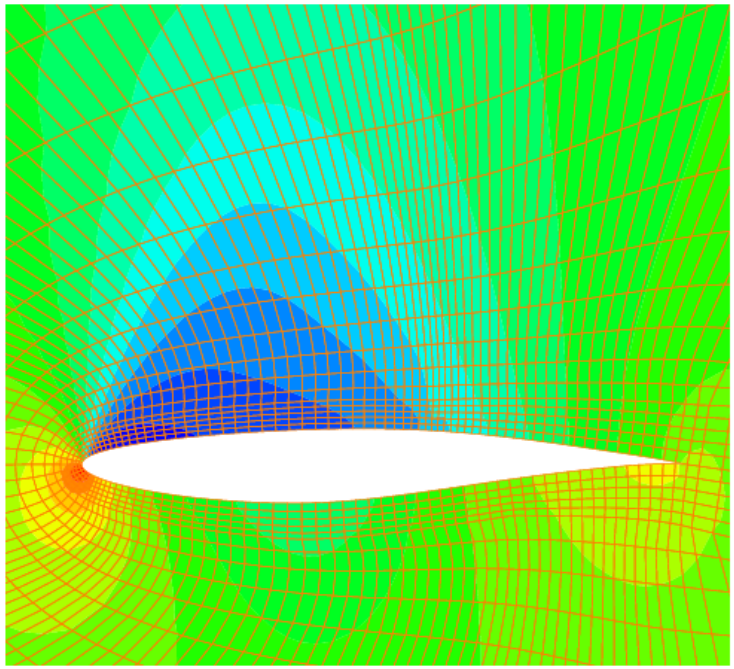
$$x_{k+1} = F(x_k, u) \qquad \text{for} \quad k = 0, 1, \ldots, N$$
$$\left(\xi_{k+1}^T, \bar{u}_{k+1}^T\right) = \xi_k^T F'(x_{N+1}, u) + \left(\bar{x}^T, 0^T\right) \qquad \text{for} \quad k = 0, 1, \ldots, \bar{N}$$

one has if $\|F'(x_{N+1}, u)\| < 1$ *that*

- *Derivatives converge*
- *Convergence rate of derivatives* $\approx$ *convergence rate of state*

Shape Optimization in Aerodynamics

- Cooperation UPB and DLR (N. Kroll, N. Gauger)
- Optimization of complete design chain
- Minimize drag s.t. lift constant

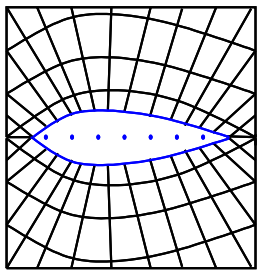


[Gauger, Walther, Özkaya, Moldenhauer 2012]

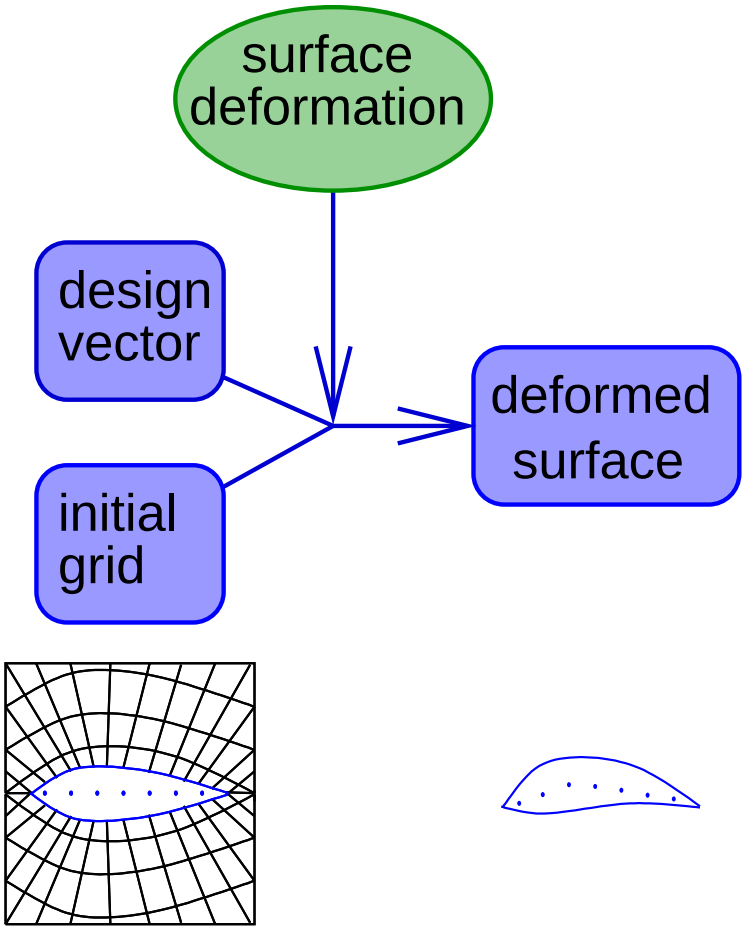
Complete Design Chain

design vector

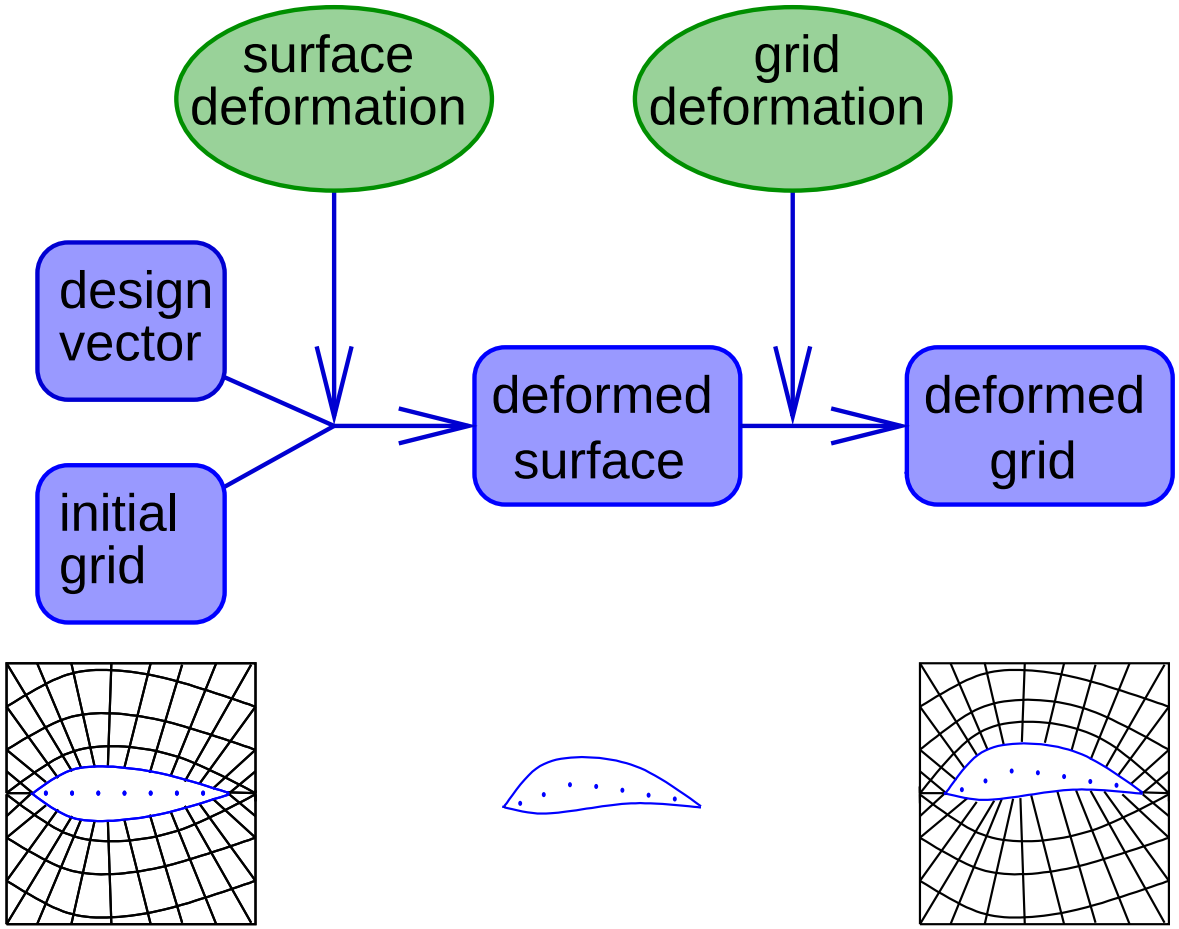
initial grid



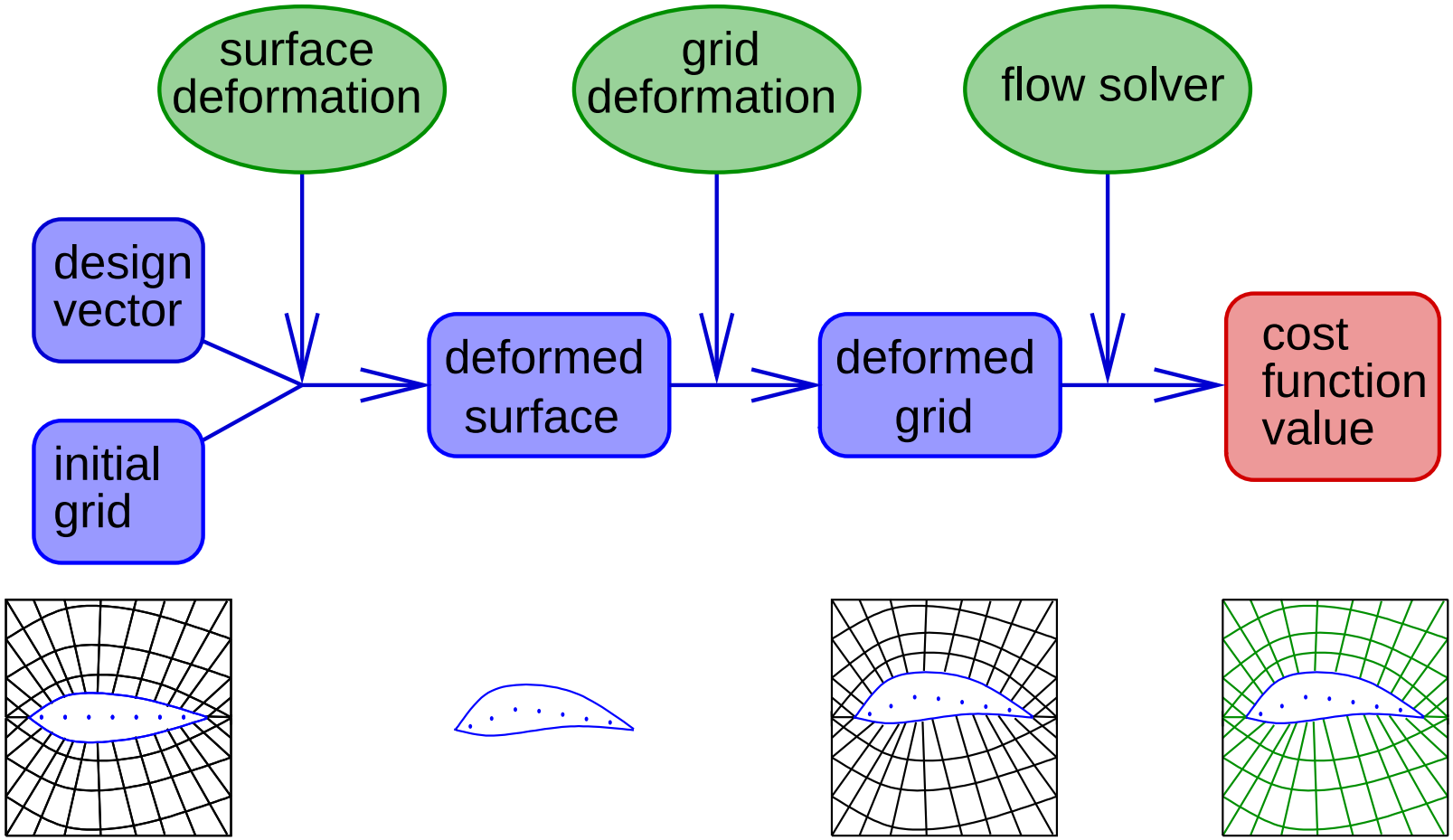
Complete Design Chain



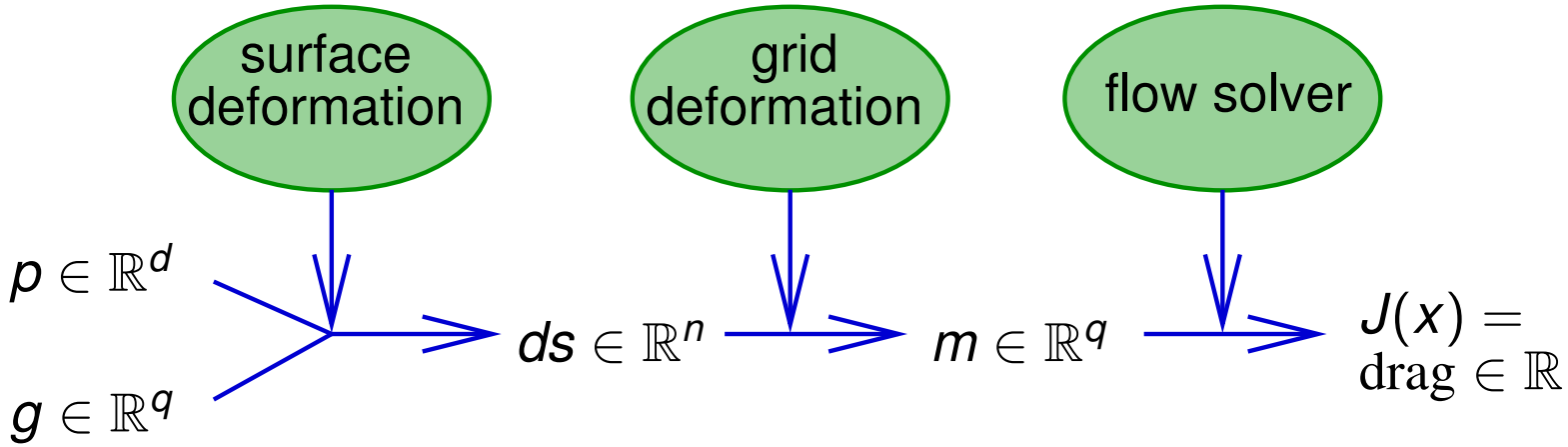
Complete Design Chain



Complete Design Chain

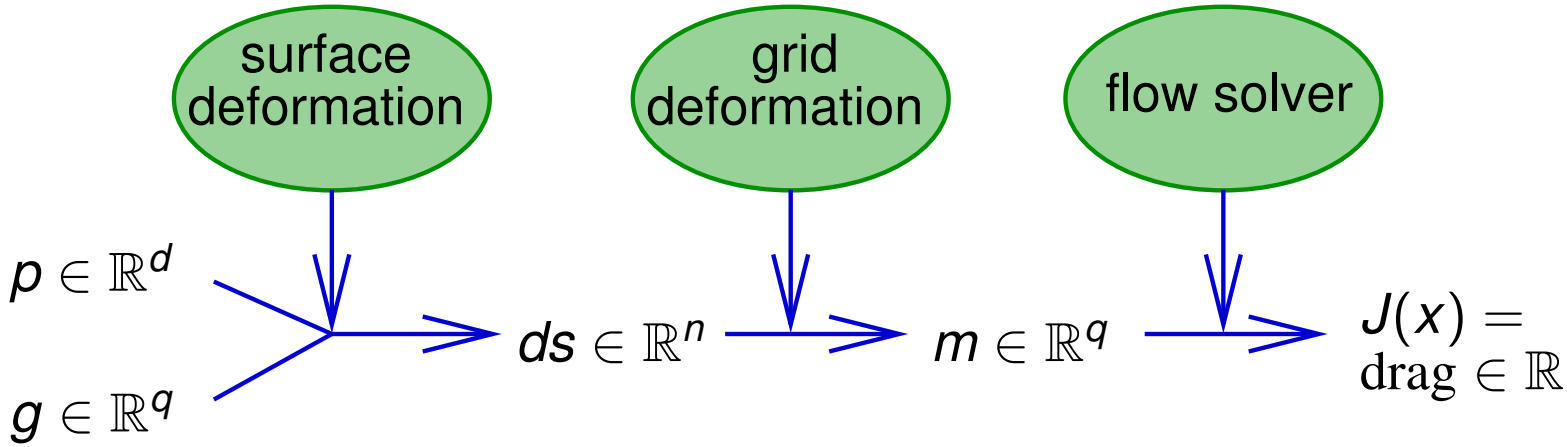


Complete Design Chain



$x \in \mathbb{R}^p$ flow, $d = 20$, n, p, q LARGE

Complete Design Chain

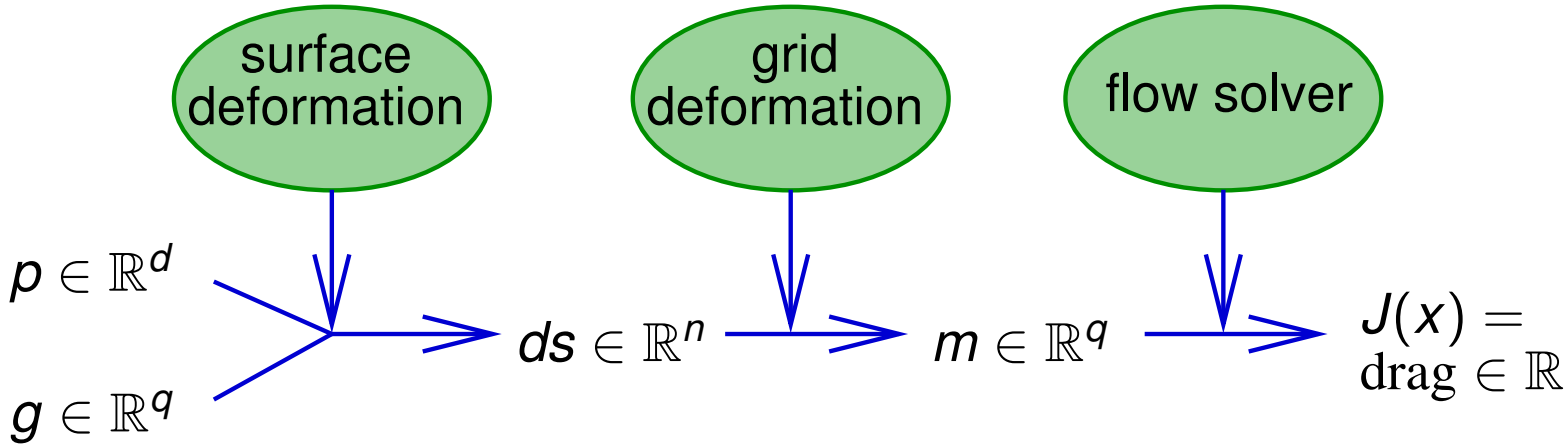


$x \in \mathbb{R}^p$ flow, $d = 20$, n, p, q LARGE

$$\frac{\partial\,\text{drag}}{\partial\,p} = \frac{\partial\,\text{drag}}{\partial\,x} \cdot \frac{\partial\,x}{\partial\,m} \cdot \frac{\partial\,m}{\partial\,ds} \cdot \frac{\partial\,ds}{\partial\,p}$$

$$\in \quad \mathbb{R}^{1\times p} \qquad \mathbb{R}^{p\times q} \qquad \mathbb{R}^{q\times n} \qquad \mathbb{R}^{n\times 20}$$

Complete Design Chain



$x \in \mathbb{R}^p$ flow, $d = 20$, n, p, q LARGE

$$\frac{\partial \,\text{drag}}{\partial p} = \frac{\partial \,\text{drag}}{\partial x} \cdot \frac{\partial x}{\partial m} \cdot \frac{\partial m}{\partial ds} \cdot \frac{\partial ds}{\partial p}$$

$$\in \quad \in \quad \in \quad \in$$

$$\mathbb{R}^{1\times p} \quad \mathbb{R}^{p\times q} \quad \mathbb{R}^{q\times n} \quad \mathbb{R}^{n\times 20}$$

Flow Solver

Euler equations for domain $\Omega \in \mathbb{R}^2$ + pseudo time:

$$\frac{\partial \omega}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0$$

$$(\omega, f, g)(t, x, y) = \begin{pmatrix} \rho & \rho u & \rho v \\ \rho u & \rho u^2 + p & \rho uv \\ \rho v & \rho uv & \rho v^2 + p \\ \rho E & \rho uH & \rho vH \end{pmatrix}$$

$\rho =$ density, $v, u =$ velocity, $E =$ energy, $H =$ enthalpy

$$p = \text{pressure} \quad \Rightarrow \quad C_p := \frac{2(p - p_\infty)}{\gamma M_\infty^2 p_\infty} \quad \Rightarrow \quad C_D$$

TAUij (DLR, N. Kroll, N. Gauger, R. Heinrich)

Test Problem: RAE2822

Two discretisations:

- I: 161 $\times$ 33 grid points
- II: 321 $\times$ 65 grid points

Complexity (one time step):

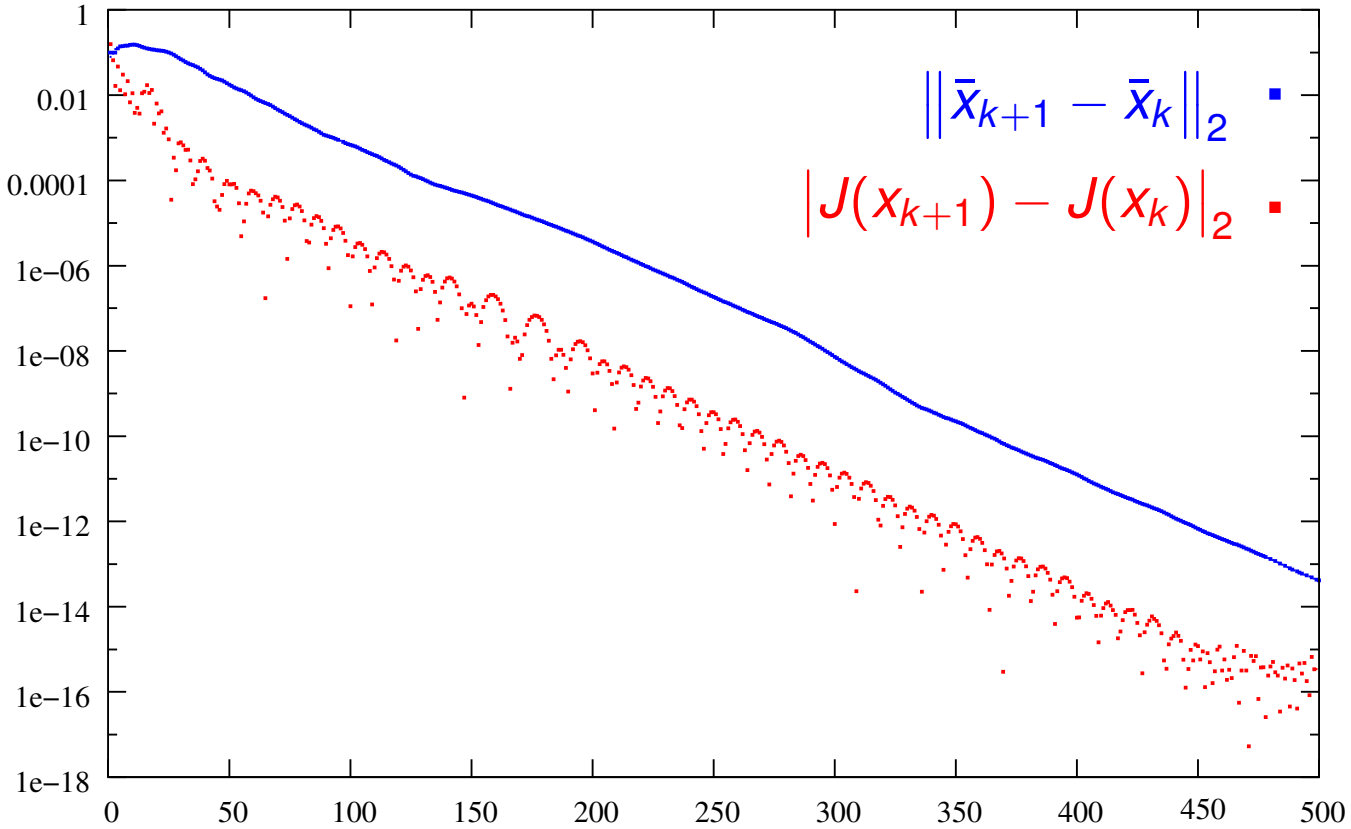
- $70 \times 10^6/285 \times 10^6$ active variables (I/II)
- $20 \times 10^6/116 \times 10^6$ operations (I/II)

Memory requirement for calculation derivatives with ADOL-C:

- 344 MB/2.3 GB for one time step (I/II)
- 2000 time steps $\Rightarrow$ “Black-Box” differentiation impossible

Convergence History (Adjoints)

- linear convergence of iterations
- same rate of convergence for both iterations



Runtimes

Discretization I	PC	cluster
2000 time steps	2 min 39 sec	1 min 44 sec
Taping	56.8 sec	2.5 sec
1000 adjoint steps	1 h 21 min 1 sec	15 min 8 sec

Diskretization II	PC	cluster
2000 time steps	11 min 36 sec	7 min 40 sec
Taping	5 min 49 sec	10.4 sec
1000 adjoint steps	$\cdots$	1 h 5 min 3 sec

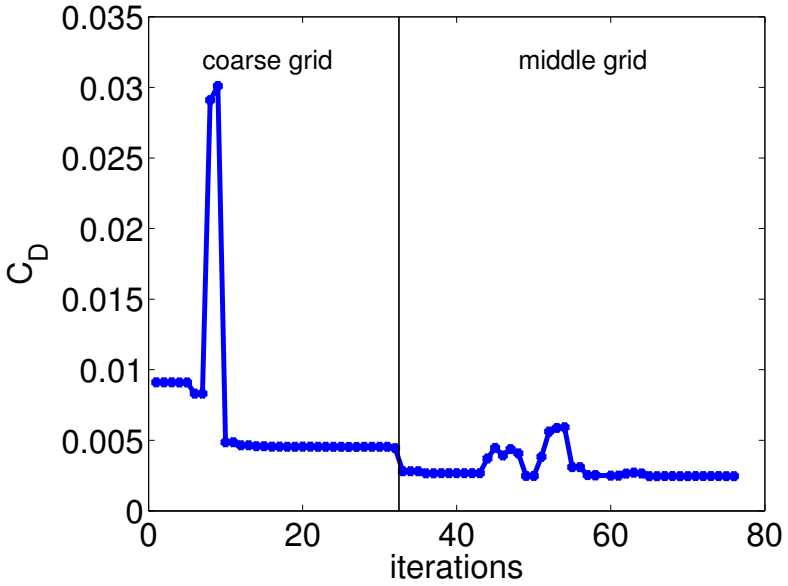
Cluster: TIME(gradient) $\leq 9 \times$ TIME(function)

PC: AMD Athlon 3200, 1 GB RAM
Cluster: single node, Dual AMD Opteron 240 (1.4GHz), 12 GB RAM

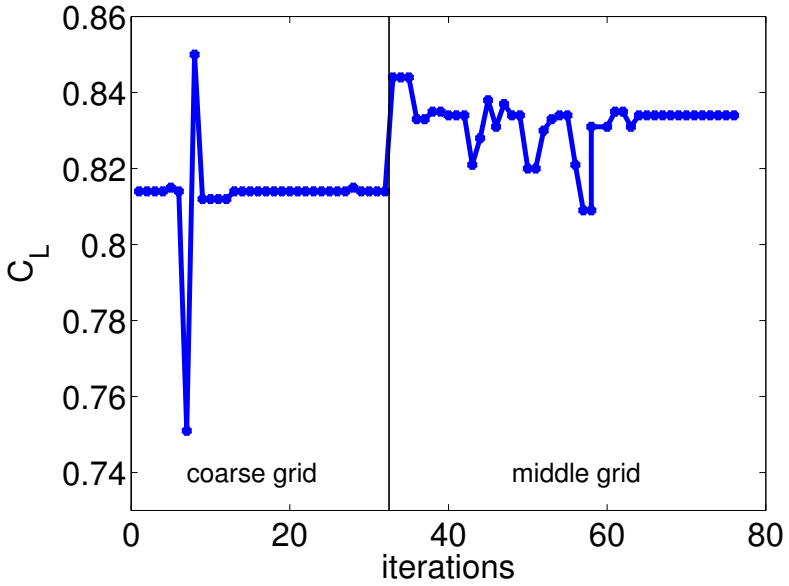
[Schlenkrich, Walther, Gauger, Heinrich 2008]

Optimization Results

development of drag



development of lift



using IPOPT [Gauger, Walther, Özkaya, Moldenhauer 2012]

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!
- Time-dependent problem:
  Implicit time-stepping ? Unknown number of time steps?
  Apply adapted checkpointing strategy

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!
- Time-dependent problem:
  Implicit time-stepping ? Unknown number of time steps?
  Apply adapted checkpointing strategy
- Pseudo time-stepping:
  Two-phase approach “easy” to apply
  One-shot approach more efficient but also more complicated

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!
- Time-dependent problem:
  Implicit time-stepping ? Unknown number of time steps?
  Apply adapted checkpointing strategy
- Pseudo time-stepping:
  Two-phase approach “easy” to apply
  One-shot approach more efficient but also more complicated
- Differentiation of Newton-like iterations OK
- Be careful when differentiating solvers like CG, GMRES, . . .