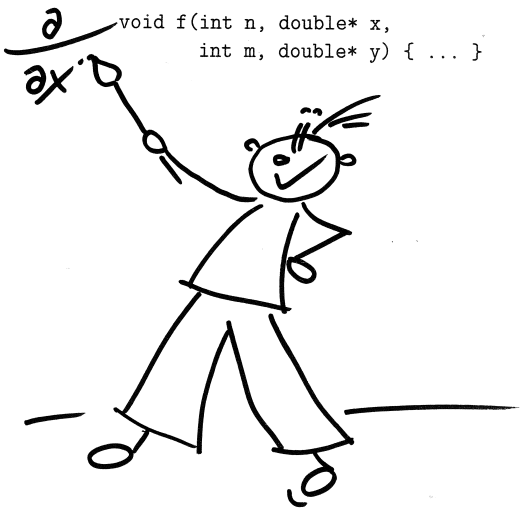


Mind the Gap in the Chain Rule

Software Tool Support for Advanced Algorithmic Differentiation of Numerical Simulation Code

Uwe Naumann

Software and Tools for Computational Engineering,
RWTH Aachen University

Graz, 12. Mar. 2015

Consider the following implementation of the scalar multivariate function $f : \mathbb{R}^n \to \mathbb{R}$

$$y = f(\mathbf{x}) = \left(\sum_{i=0}^{n-1} x_i^2 \right)^2$$

in C++:

```
template<typename T>
void f(const vector<T> &x, T &y) {
  y=0;
  for (size_t i=0;i<x.size();i++) y+=x[i]*x[i];
  y*=y;
}
```

We are interested in the gradient $f'(\mathbf{x})$ at point $\mathbf{x} = (\cos(i))_{i=0,\ldots,n-1}$ and for $n = 10^5$. Let us try central finite differences ...

Motivation

Algorithmic Differentiation (AD)

- 1st-Order ($F : \mathbb{R}^n \to \mathbb{R}^m$)
  - Tangents: $\boldsymbol{y}^{(1)} := \nabla F(\mathbf{x}) \cdot \mathbf{x}^{(1)}$
  - Adjoints: $\mathbf{x}_{(1)} := \mathbf{x}_{(1)} + \nabla F(\mathbf{x})^T \cdot \boldsymbol{y}_{(1)}$

- 2nd-Order (w.l.o.g. $f : \mathbb{R}^n \to \mathbb{R}$)
  - Tangents:

$$y^{(1,2)} := \mathbf{x}^{(1)^T} \cdot \nabla^2 f(\mathbf{x}) \cdot \mathbf{x}^{(2)} + \nabla f(\mathbf{x}) \cdot \mathbf{x}^{(1,2)}$$

  - Adjoints:

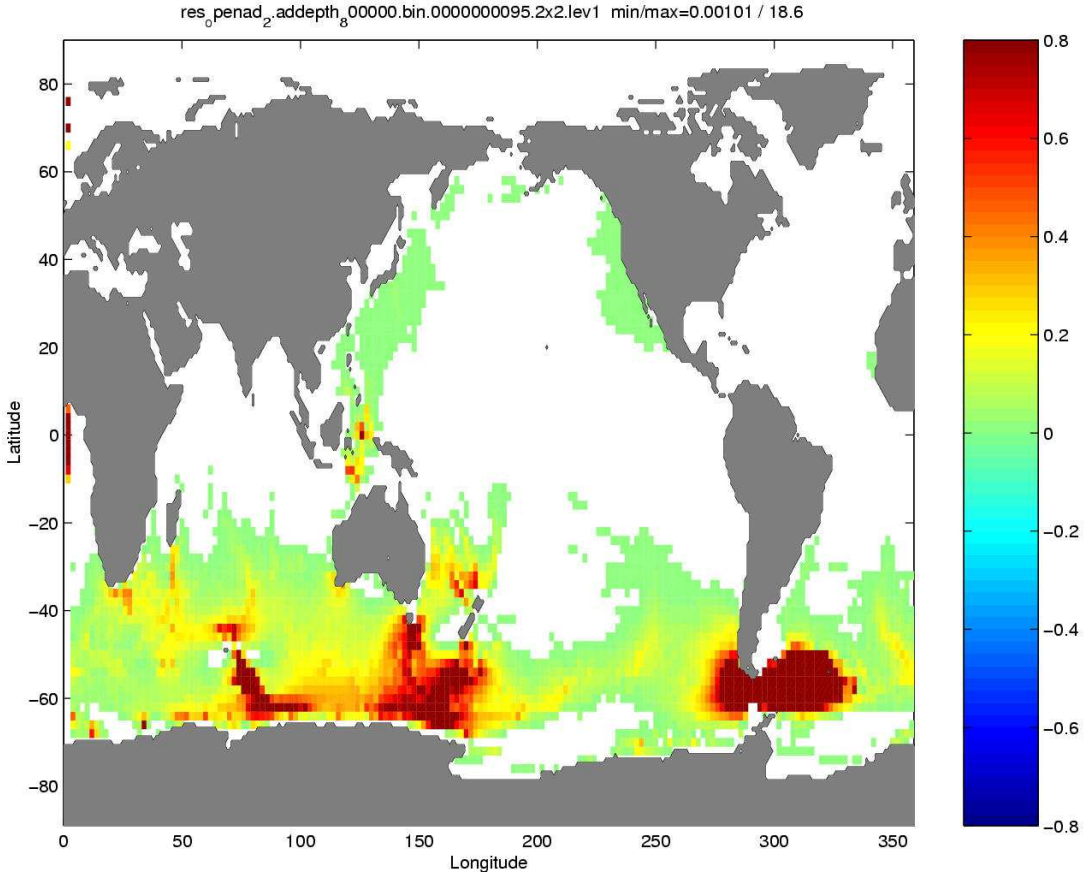
$$\mathbf{x}^{(2)}_{(1)} := \mathbf{x}^{(2)}_{(1)} + y_{(1)} \cdot \nabla^2 f(\mathbf{x}) \cdot \mathbf{x}^{(2)} + \nabla f(\mathbf{x})^T \cdot y^{(2)}_{(1)}$$

(©J.Lotz)

- 3rd-Order ...
- ...

Motivation

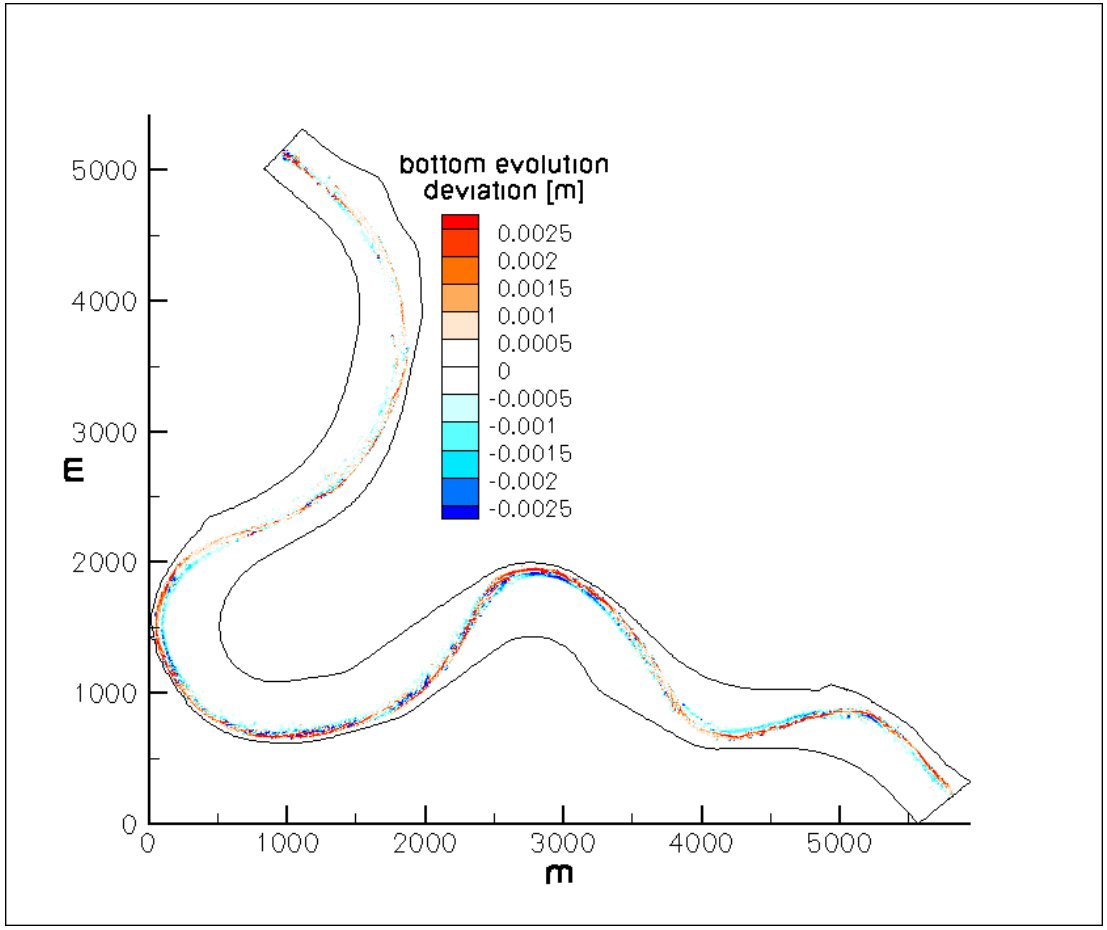
Sensitivities in Applications: Physical Oceanography



Collab. with EAPS @ MIT

Motivation

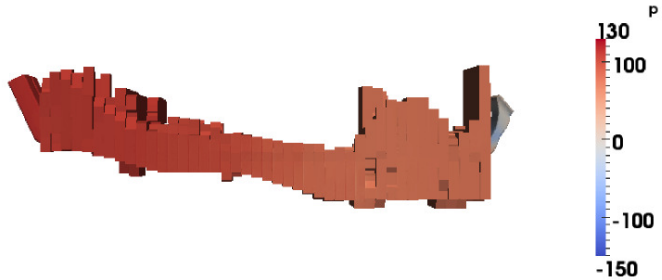
Sensitivities in Applications: Waterways Engineering

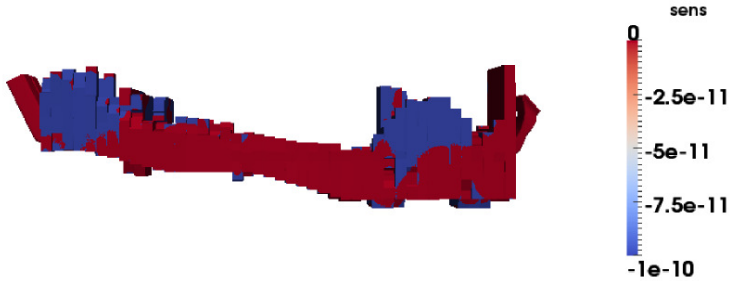


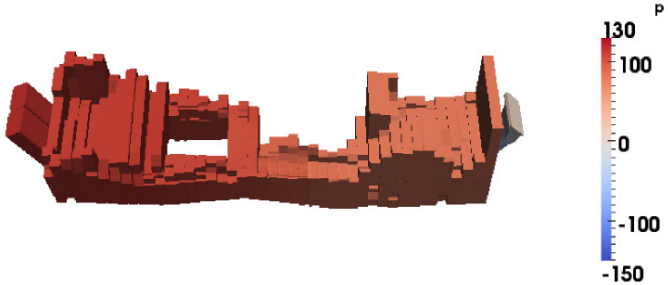
Collab. with BAW

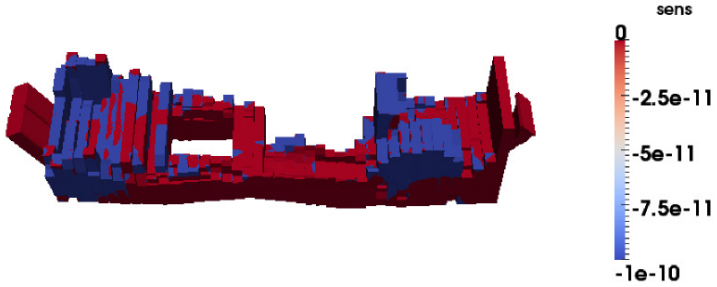
Motivation

Sensitivities in Applications: Automotive Engineering





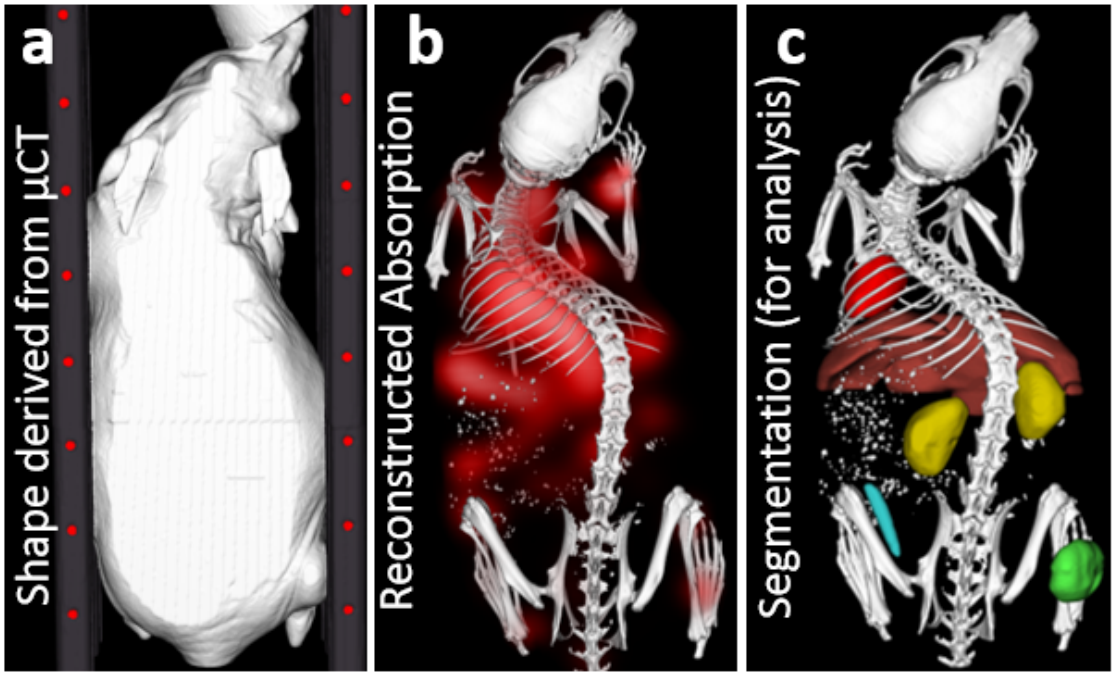


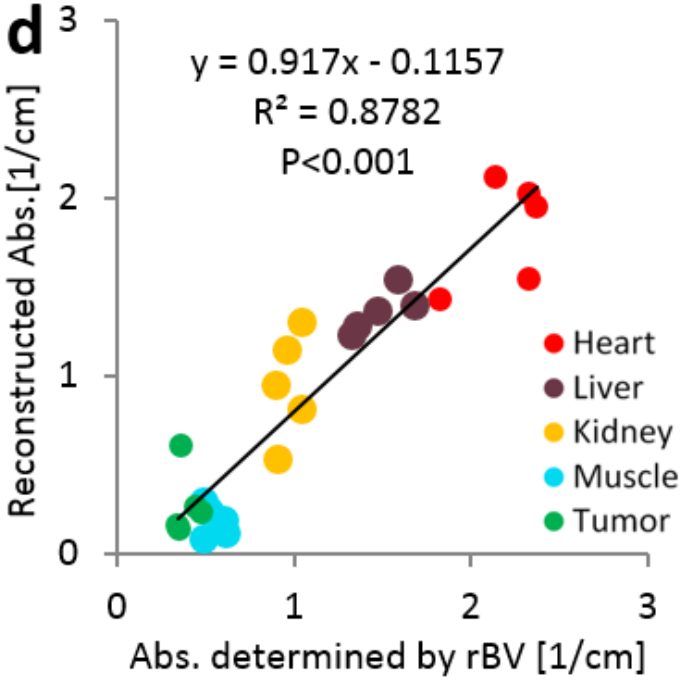


Collab. with Volkswagen AG

Motivation

Sensitivities in Applications: Medical Imaging





Collab. with ExMI @ RWTH

Motivation

Results

1. central finite difference (CFD) approximation takes $\approx 85s$
   (`time ./main.exe 100000 > cfd.out`)
2. tangent algorithmic differentiation (TAD) takes $\approx 77s$
   (`time ./main.exe 100000 > gt1s.out`)
3. adjoint algorithmic differentiation (AAD) takes $\approx 0.5s$
   (`time ./main.exe 100000 > ga1s.out`)
4. TAD and AAD produce identical numerical results up to machine precision
   (`gvimdiff gt1s.out ga1s.out`)
5. CFD produces a rather poor approximation of the gradient
   (`gvimdiff cfd.out ga1s.out`)
6. AAD runs out of memory for $n = 10^8$ if swapping to disc disabled

Outline

Outline

Let

$$\mathbb{R}^m \supseteq I \ni \boldsymbol{y} = F(\mathbf{x}) = (F_l \circ \ldots \circ F_1)(\mathbf{x})$$

be continuously differentiable over the domain $D \subseteq \mathbb{R}^n$ and let it be decomposable into a sequence of elemental function evaluations $\mathbf{x}_i = F_i(\mathbf{x}_{i-1})$, $i = 1, \ldots, l$, such that $\mathbf{x} = \mathbf{x}_0$ and $\boldsymbol{y} = \mathbf{x}_l$.

Then both the Jacobian

$$\nabla F = \nabla F_l \cdot \ldots \cdot \nabla F_1$$

and each of the local Jacobians $\nabla F_i = \nabla F_i(\mathbf{x}_{i-1})$, $i = 1, \ldots, l$, exist and are finite over their respective domains.

Tangent AD computes directional derivatives

$$\dot{\boldsymbol{y}} = \nabla F \cdot \dot{\mathbf{x}} = \nabla F_i \cdot (\ldots \cdot (\nabla F_1 \cdot \dot{\mathbf{x}}) \ldots) \quad .$$

Mind the Gap

Adjoint AD

Adjoint AD computes $\bar{\mathbf{x}} = \nabla F^T \cdot \bar{\boldsymbol{y}} = \nabla F_1^T \cdot (\ldots(\nabla F_l^T \cdot \bar{\mathbf{x}}_l)\ldots)$ assuming availability of adjoint elementals

$$\bar{\mathbf{x}}_{i-1} = \bar{F}_i(\bar{\mathbf{x}}_i) \equiv \nabla F_i^T \cdot \bar{\mathbf{x}}_i \tag{1}$$

for $i = l, \ldots, 1$ (mind reversal of data flow). AD tools implement (1) for the intrinsic operations $(+, *, \ldots)$ and functions $(\sin, \exp, \ldots)$ of the respective programming language.

Let $\bar{\mathbf{x}}_{k-1} = \bar{F}_k(\bar{\mathbf{x}}_k)$ supposed to be treated differently, that is, not by standard AD. From the AD tool's perspective this situation induces a gap in the chained Jacobian product

$$\bar{\mathbf{x}} = \nabla F^T \cdot \bar{\boldsymbol{y}} = \nabla F_1^T \cdot \ldots \cdot \overbrace{\nabla F_k^T(\mathbf{x}_{k-1}) \cdot \underbrace{(\nabla F_{k+1}^T \cdot \ldots \cdot (\nabla F_l^T \cdot \bar{\mathbf{x}}_l)\ldots)}_{\bar{\mathbf{x}}_k}}^{\bar{\mathbf{x}}_{k-1}}$$

to be filled by a version of $\bar{F}_k$ provided by the user.

Mind the Gap
Scenarios

- checkpointing
- preaccumulation
- parallelism
- symbolic adjoints of implicit functions
- user-supplied (approximate) adjoint code
- smoothing
- (domain-specific) adjoint library routines

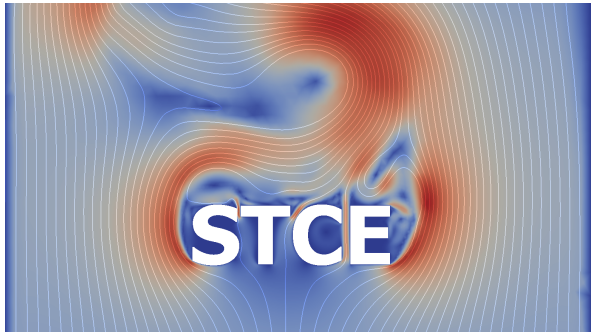
... to be refined later.

Outline

STCE @ RWTH

Fact Sheet

- founded 2004
- based at IT Center
- 12 postgraduate r&d, 5 undergraduate r&d, administrative support through IT Center
- 5 PhDs completed; expected in 2015 (2), 2016 (3)
- member of Aachen Institute for Advanced Studies in Computational Engineering Science (CES)
- Computer Science in Ba/Ma CES; mentoring

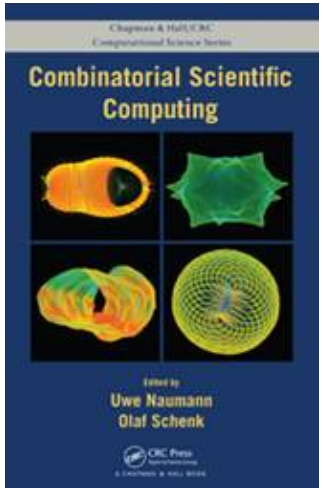


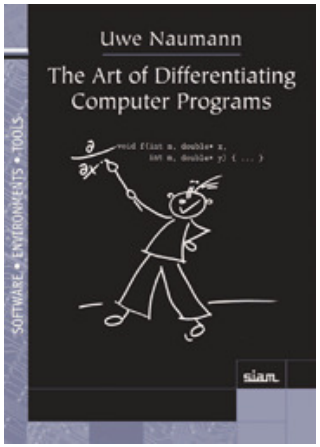
©M. Towara using OpenFOAM

STCE @ RWTH

R&D / Teaching

- R&D
  - software tool support for discrete adjoint numerical methods
  - algorithmic differentiation
  - simulation program analysis and transformation
  - parallel programming
  - combinatorial scientific computing
  - applications in science, engineering, and finance





- Teaching
  - programming (C++, Fortran, MPI, OpenMP)
  - simulation software engineering
  - computational differentiation
  - numerical program transformation
  - combinatorial problems in scientific computing

- dco/c++
- dco/fortran
- NAG AD compiler (preprocessor for dco/fortran)
- dcc (OpenMP support)
- discrete adjoint OpenFOAM / OpenFOAM Extend
- ADE (Aachen Dynamic Optimization Environment)
- AMPI (adjoint MPI)
- modMC (adjoint subgradients for McCormick relaxations)
- ctr (optimized call tree reversals by IP)
- NAG AD Library

https://www.stce.rwth-aachen.de/software/index.html

Outline

AD Software dco/c++

- generic first- and higher-order tangents and adjoints
- multi-threaded vector modes
- tape types: blob, chunk, multiple, file, MPI
- tape compression through preaccumulation / meta-programming
- generic external adjoint object interface
- vectorized intrinsics for GPU (CUDA) [5]
- detection and exploitation of sparsity [8]
- NAG AD Library support
- AMPI support [20]
- PETSc support [13]
- Python support (work in progress)
- actively used by BAW, DLR, EDF, FZJ, MPI-M, ... and various tier-1 investment banks

https://www.stce.rwth-aachen.de/software/dco_cpp.html

AD Software dco/c++

First-Order Tangents: $\boldsymbol{y}^{(1)} = \nabla F(\mathbf{x}) \cdot \mathbf{x}^{(1)}$

```
#include <vector>
#include "dco.hpp" // C++ template library
using namespace dco;

template<typename T>
void f(const std::vector<T>& x, std::vector<T>& y); // generic primal

void gt1s_driver( const std::vector<double>& xv,
                  const std::vector<double>& xt,
                  std::vector<double>& yv,
                  std::vector<double>& yt) {
  int n=xv.size(), m=yv.size(); std::vector<gt1s<double>::type> x(n), y(m);
  for (int i=0;i<n;i++) { // seeding
    value(x[i])=xv[i];
    derivative(x[i])=xt[i];
  }
  f(x,y); // overloaded primal run
  for (int i=0;i<m;i++) { // harvesting
    yv[i]=value(y[i]);
    yt[i]=derivative(y[i]);
  }
}
```

Recall: Adjoint AD computes

$$\bar{\mathbf{x}} = \nabla F^T \cdot \bar{\boldsymbol{y}} = \nabla F_1^T \cdot \ldots \cdot \overbrace{\nabla F_k^T(\mathbf{x}_{k-1}) \cdot \underbrace{(\nabla F_{k+1}^T \cdot \ldots \cdot (\nabla F_l^T \cdot \bar{\mathbf{x}}_l) \ldots)}_{\bar{\mathbf{x}}_k}}^{\bar{\mathbf{x}}_{k-1}} \tag{2}$$

assuming availability of adjoint elementals $\bar{\mathbf{x}}_{i-1} = \bar{F}_i(\bar{\mathbf{x}}_i) \equiv \nabla F_i^T \cdot \bar{\mathbf{x}}_i$ *for* $i \in \{1, \ldots, l\}$.

An appropriately augmented version of the given implementation of F is executed to generate a so-called tape. This recording procedure is followed by the interpretation of the tape to evaluate (2).

A tape is an abstract data structure defined as a sequence of adjoint objects.

An adjoint object holds all data necessary for executing some $\bar{F}_i$. Its evaluation yields $\bar{\mathbf{x}}_{i-1}$ for given $\bar{\mathbf{x}}_i$. Interpretation of the tape amounts to the evaluation of the adjoint objects for $i = l, \ldots, 1$.

Gap $\Rightarrow$ adjoint object to be provided by user.

AD Software dco/c++

First-Order Adjoints: $\mathbf{x}_{(1)} = \nabla F(\mathbf{x})^T \cdot \boldsymbol{y}_{(1)}$

```
void ga1s_driver(const std::vector<double>& xv,
                 std::vector<double>& xa,
                 std::vector<double>& yv,
                 std::vector<double>& ya) {
  ga1s<double>::global_tape=ga1s<double>::tape_t::create(); // tape creation
  int n=xv.size(), m=yv.size();
  std::vector<ga1s<double>::type> x(n), y(m);
  for (int i=0;i<n;i++) { // registration of active inputs
    ga1s<double>::global_tape->register_variable(x[i]);
    value(x[i])=xv[i];
  }
  f(x,y); // overloaded primal run records tape
  for (int i=0;i<m;i++) {
    yv[i]=value(y[i]);
    derivative(y[i])=ya[i]; // seeding
  }
  for (int i=0;i<n;i++) derivative(x[i])=xa[i];
  ga1s<double>::global_tape->interpret_adjoint(); // tape interpretation
  for (int i=0;i<n;i++) xa[i]=derivative(x[i]); // harvesting
  ga1s<double>::tape_t::remove(ga1s<double>::global_tape);
}
```

AD Software dco/c++

Competition + Disclaimer!

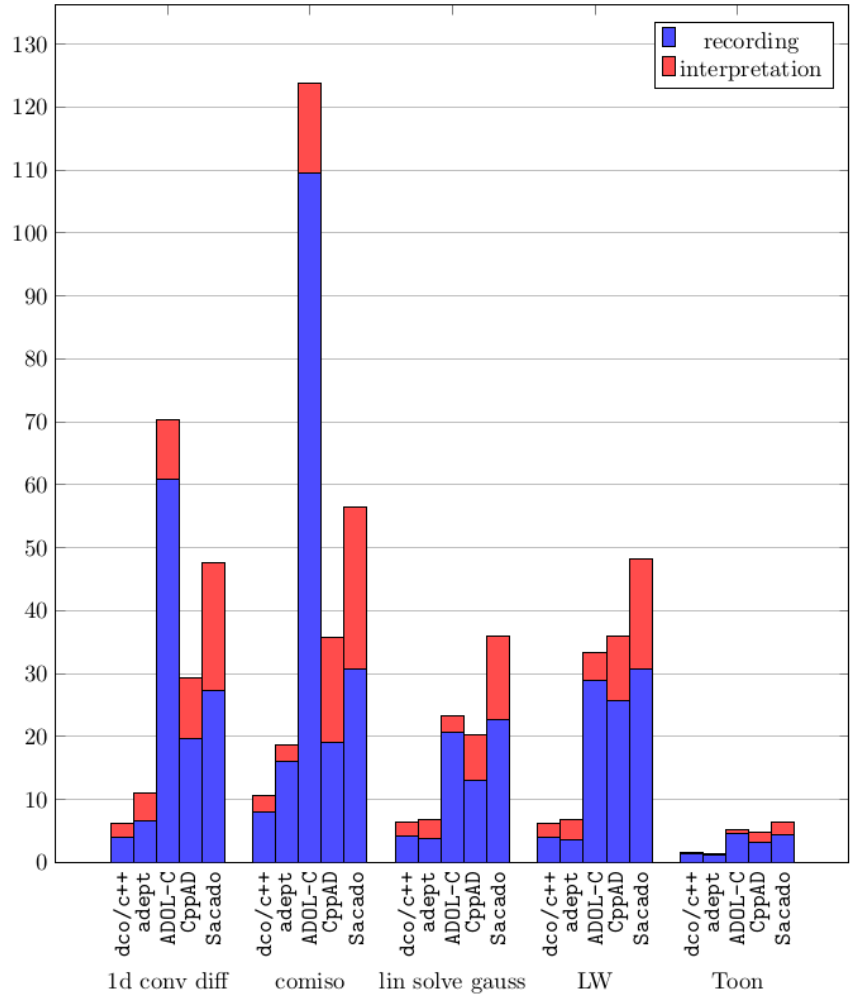


Figure 1.1: Run time Ratio

Test Problems (Black-Box)

1. explicit Euler on 1D convection-diffusion equation (J. Lotz)
2. CoMISo - Constrained Mixed-Integer Solver (D. Bommes)
3. LU decomposition (text book)
4. Lax-Wendroff scheme for hyperbolic PDEs [11]
5. Toon scheme for hyperbolic PDEs [11]

However ...

Outline

Recall: Adjoint AD computes

$$\bar{\mathbf{x}} = \nabla F^T \cdot \bar{\boldsymbol{y}} = \nabla F_1^T \cdot \ldots \cdot \overbrace{\nabla F_k^T(\mathbf{x}_{k-1}) \cdot \underbrace{(\nabla F_{k+1}^T \cdot \ldots \cdot (\nabla F_l^T \cdot \bar{\mathbf{x}}_l) \ldots)}_{\bar{\mathbf{x}}_k}}^{\bar{\mathbf{x}}_{k-1}} \tag{3}$$

assuming availability of adjoint elementals $\bar{\mathbf{x}}_{i-1} = \bar{F}_i(\bar{\mathbf{x}}_i) \equiv \nabla F_i^T \cdot \bar{\mathbf{x}}_i$ *for* $i \in \{1, \ldots, l\}$.

The user may choose to provide a custom version for one or several $\bar{F}_i$.

Consequently, the AD tool needs to meet the following requirements:

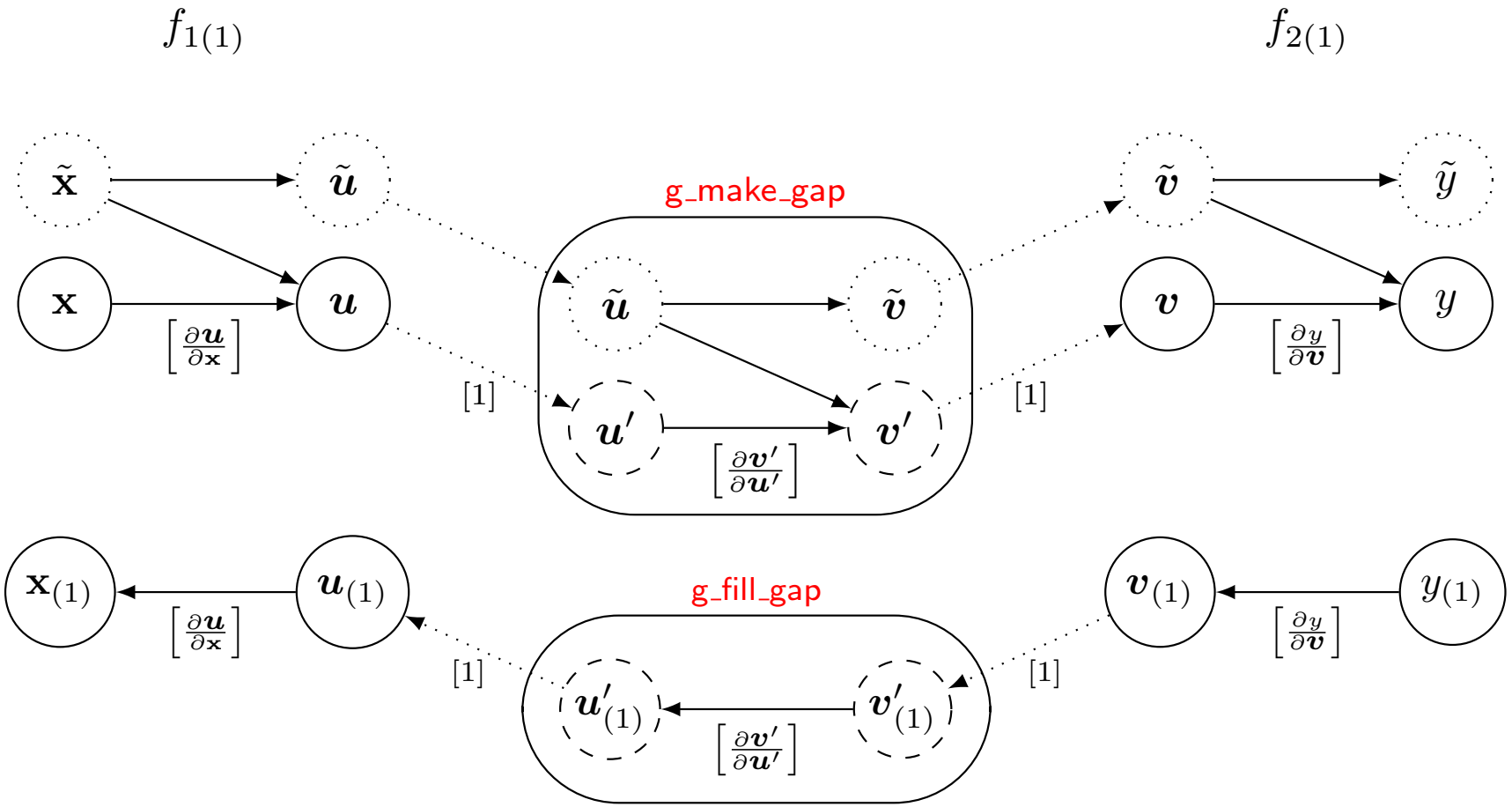
- Ability to define custom adjoint objects within the tape;
- their evaluation by the tape interpreter;
- ability to read the value of $\bar{\mathbf{x}}_i$;
- ability to write the value of $\bar{\mathbf{x}}_{i-1}$.

Data required by $\bar{F}_i$ (for example, $\mathbf{x}_{i-1}$ defining ∇F_i) needs to be collected by an appropriately modified version of F_i ($*$_make_gap).

$\bar{F}_i$ needs to be called by tape interpreter ($*$_fill_gap).

Mind the Gap $(\tilde{\boldsymbol{v}}, \boldsymbol{v}) = g(\tilde{\boldsymbol{u}}, \boldsymbol{u})$

User-Supplied Adjoint Objects



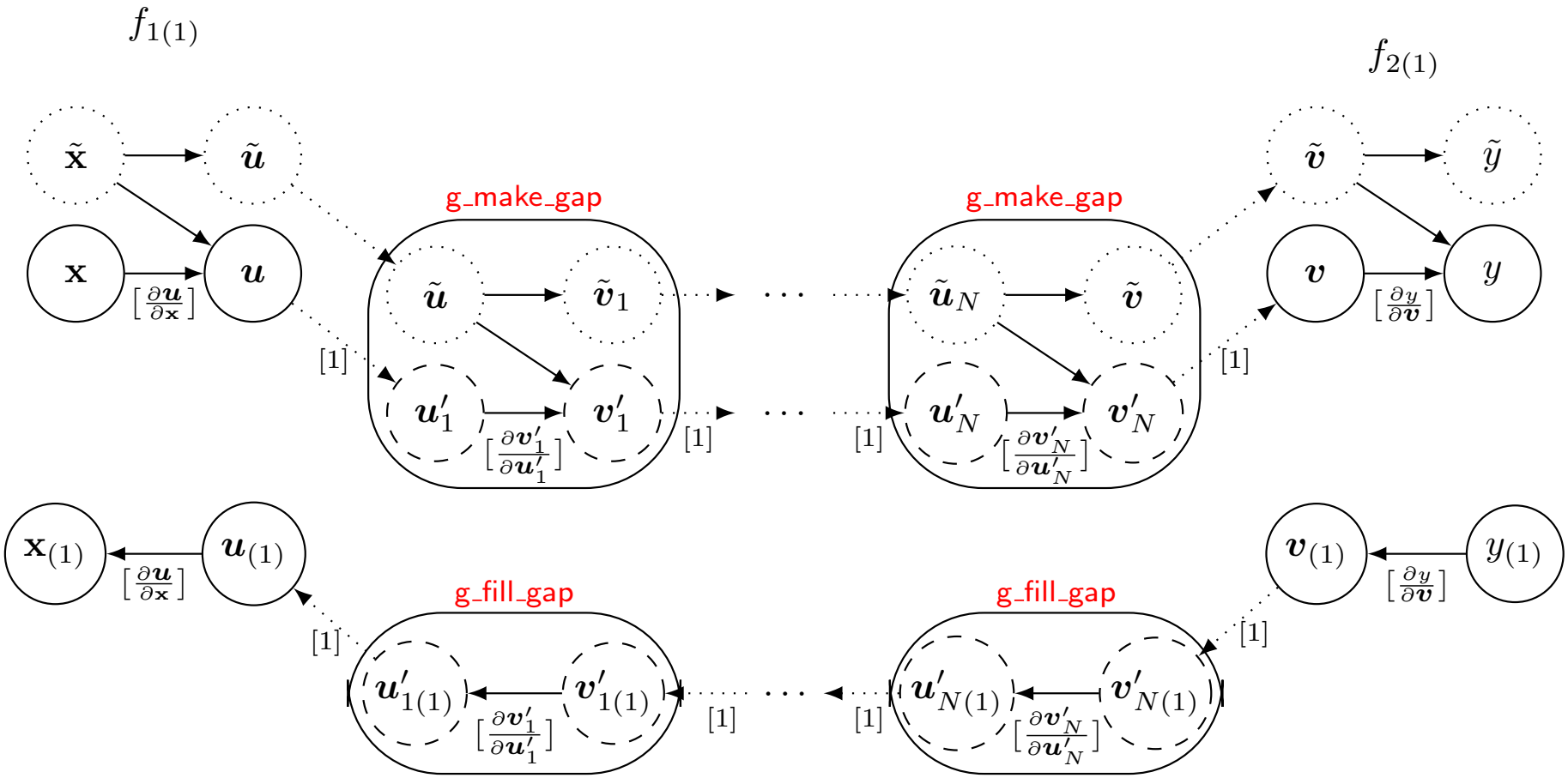
Mind the Gap

Scenarios

- checkpointing
  - call trees (subprogram calls) [14]
  - evolutions (sequential loops) [7]
  - ensembles (parallel loops) [9]
- symbolic adjoints of implicit functions
  - (non-)linear systems [4]
  - reverse accumulation [3]
  - ODEs/DAEs [19]
  - PDEs [12]
- user-supplied (approximate) adjoint code
  - hand-written adjoint code
  - use of accelerators (GPUs,...) [5]
  - combination with source transformation AD tools [10]
  - finite differences (e.g. black boxes, smoothing)
- (domain-specific) adjoint libraries
  - adjoint GSL
  - adjoint NAG Library
- ...

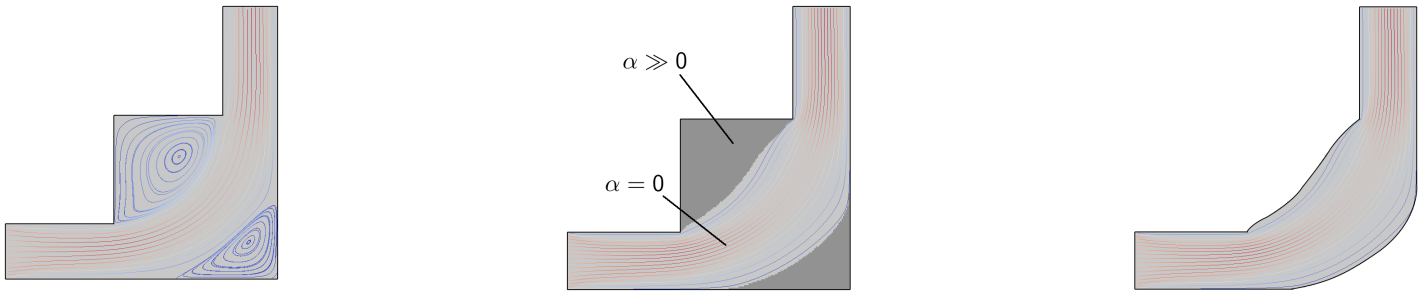
Mind the Gap

Checkpointing Evolutions

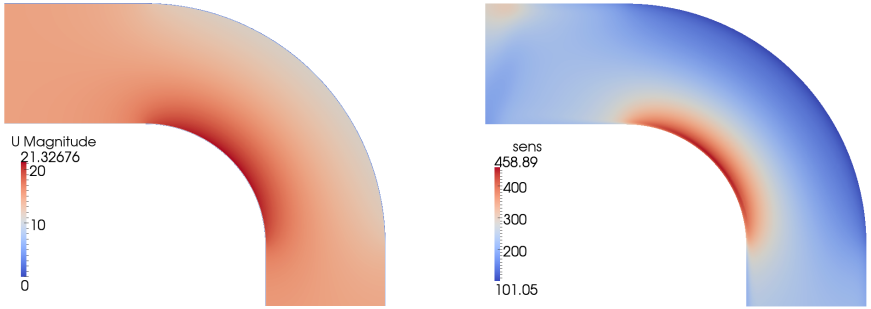


- idea: impermeability parameter $\alpha \in \mathbb{R}$

initial geometry | optimized flow | optimized geometry



- case study [22] (723.000 cells $\rightarrow \boldsymbol{\alpha} \in \mathbb{R}^{723.0000}$)



$\rightarrow$ M. Towara, A. Sen, A. Dastouri

We minimize total dissipation

$$J(\alpha) = -\int_\Gamma \left(p + \frac{1}{2}\mathbf{v}^2\right) \mathbf{v} \cdot \mathbf{n} \, \mathrm{d}\Gamma$$

over the in-/outlet Γ of a duct with the flow described by the kD Navier-Stokes equations

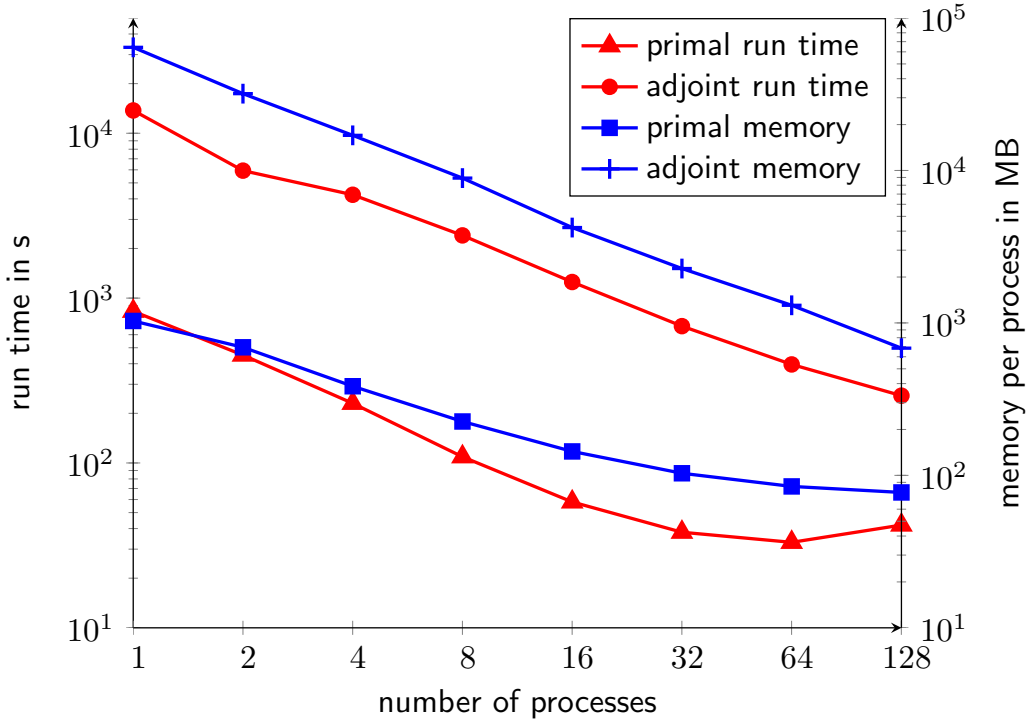
$$\mathbf{v} \cdot \nabla \mathbf{v} = \nu \nabla^2 \mathbf{v} - \frac{1}{\rho} \nabla p - \alpha \mathbf{v}$$
$$\nabla \cdot \mathbf{v} = 0$$

with an additional penality term depending on impermeability α and where $\mathbf{v} \in \mathbb{R}^k$ and $\nu, \rho, p, \alpha \in \mathbb{R}$ [17]. The gradient of the objective J with respect the discrete $\boldsymbol{\alpha}$ required, for example, by a gradient descent optimization method

$$\boldsymbol{\alpha}^{n+1} = \boldsymbol{\alpha}^n - \lambda \cdot \frac{\partial J}{\partial \boldsymbol{\alpha}^n}$$

with line search parameter λ is computed by Discrete Adjoint OpenFOAM [21].

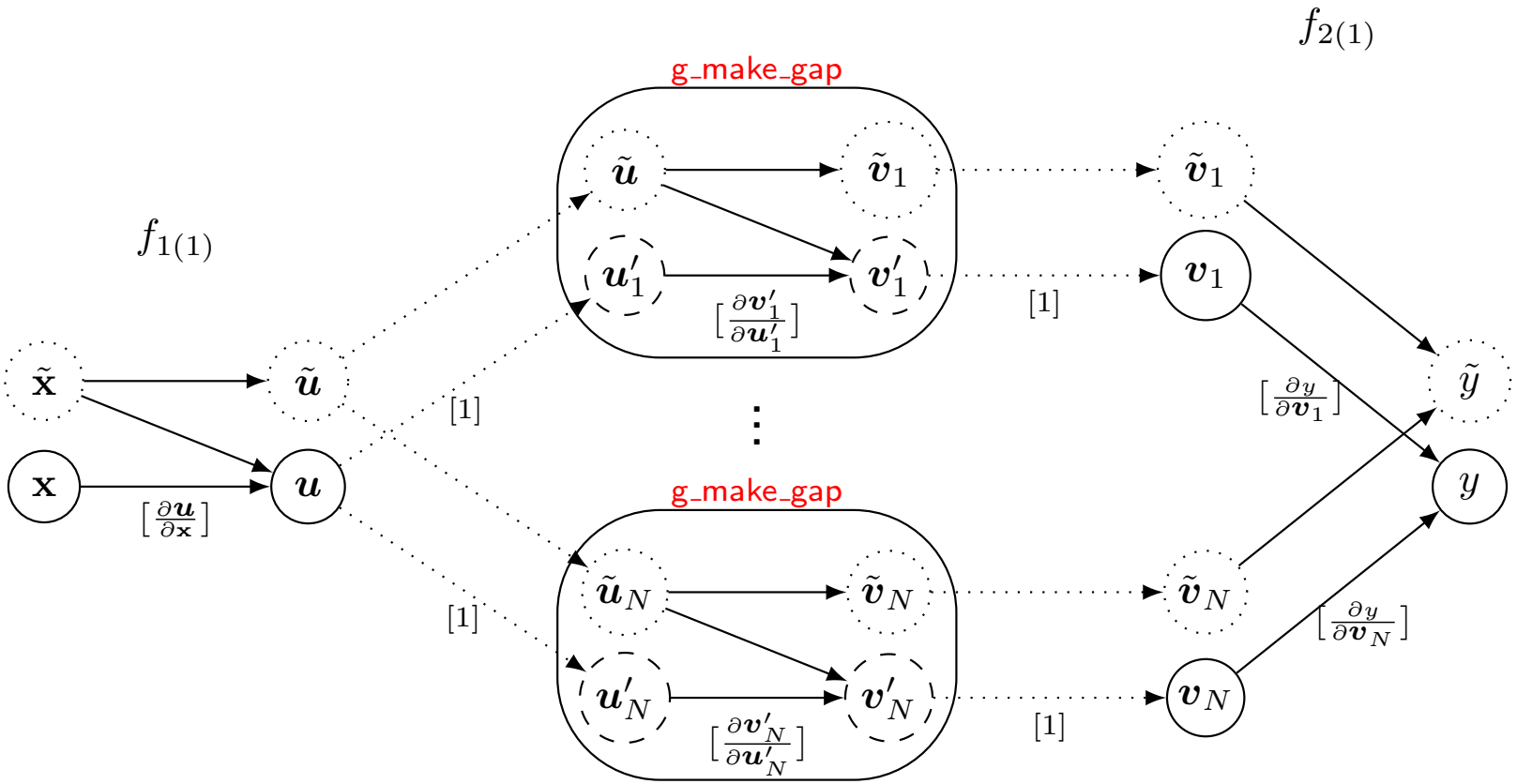
The Navier-Stokes equations are solved to convergence by 300 SIMPLE [18] iterations. The dco/c++ tape of a single iteration occupies ~ 60GB yielding an infeasible total memory requirement of ~ 18TB. Binomial checkpointing [7] reduces the total memory requirement to ~ 65GB. Distributed memory parallelization with MPI exhibits promising scaling behavior.



The relative computational cost of the adjoint is ~ 15.

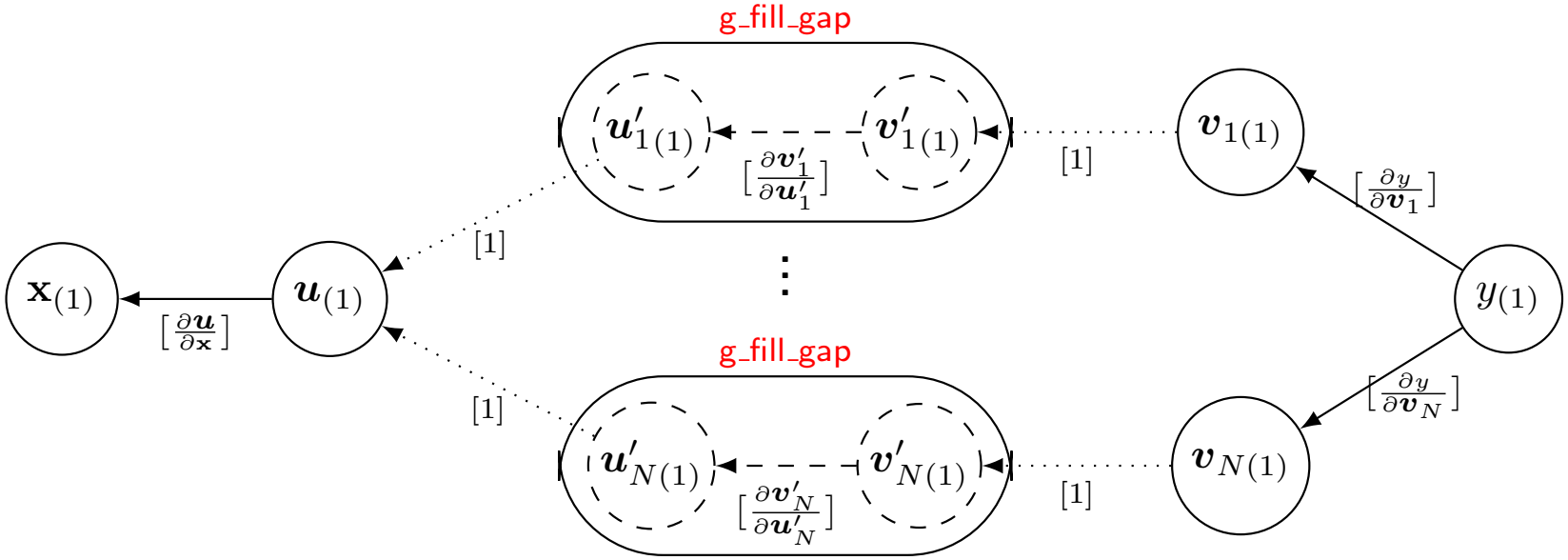
Mind the Gap

Checkpointing Ensembles



Mind the Gap

Checkpointing Ensembles



We price a simple European Call option written on an underlying $S = S(t) : \mathbb{R}^+ \to \mathbb{R}^+$, described by the SDE

$$dS(t) = S(t) \cdot r \cdot dt + S(t) \cdot \sigma(t, S(t)) \cdot dW(t)$$

with time $t \geq 0$, maturity $T > 0$, strike $K > 0$, constant interest rate $r > 0$, volatility $\sigma = \sigma(t, S) : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$, and Brownian motion $W = W(t) : \mathbb{R}^+ \to \mathbb{R}^+$.

The value of a European call option driven by S is given by the expectation

$$V = \mathbb{E} e^{-r \cdot T} \cdot (S(T) - K)^+$$

for given interest r, strike K, and maturity T. It can be evaluated using Monte Carlo simulation; e.g, 360 Euler steps per 10^5 paths ...

$\to$ J. Deussen, V. Mosenkis

Results [15]

n	primal	cfd	a1s	a1s_ensemble	$\mathcal{R}$
10	0.3s	6.1s	1.8s (2GB)	1.3s (1.9MB)	4.3
22	0.4s	15.7s	- ($>$ 3GB)	2.3s (2.2MB)	5.7
34	0.5s	29.0s	- ($>$ 3GB)	3.0s (2.5MB)	6.0
62	0.7s	80.9s	- ($>$ 3GB)	5.1s (3.2MB)	7.3
142	1.5s	423.5s	- ($>$ 3GB)	12.4s (5.1MB)	8.3
222	2.3s	1010.7s	- ($>$ 3GB)	24.4s (7.1MB)	10.6

i	ffd	cfd	t1s	a1s_ensemble
0	0.98209703	0.982097084	**0.98209708315948**	**0.98209708315948**
7	-0.07167053	-0.07166695	**-0.07166605682464**	**-0.07166605682464**
4	0.34617424	0.346131324	**0.34612682023931**	**0.34612682023932**

Mind the Gap

Symbolic Adjoints (of Implicit Functions)

$$
\begin{tikzcd}
\boxed{x^2 - p = 0} \arrow[r, "x_{(1)} \cdot \frac{d}{dp}"] \arrow[d, red, "x^* = S(x^0, p)"', "\text{record}"] & \boxed{x_{(1)} \cdot \left(2 \cdot x \cdot \frac{dx}{dp} - 1\right) = 2 \cdot x \cdot x_{(1)} \cdot \frac{dx}{dp} - x_{(1)} = 2 \cdot x \cdot p_{(1)} - x_{(1)} = 0} \arrow[d, blue, "\tilde{p}_{(1)} = \tilde{S}(p^0_{(1)}, x^*, x_{(1)})"] \\
\boxed{x^* \approx \sqrt{p}} \arrow[r, red, "\text{interpret}", "\bar{p}_{(1)} = S_{(1)}(x^0, x_{(1)}, p)"'] & \boxed{p_{(1)} = p_{(1)} + x_{(1)} \cdot \frac{dx}{dp} = \frac{x_{(1)}}{2 \cdot \sqrt{p}} = \frac{x_{(1)}}{2 \cdot x} \approx \bar{p}_{(1)} \approx \tilde{p}_{(1)}}
\end{tikzcd}
$$

Similar ideas apply to systems of linear [4] and nonlinear [6] equations.

For Newton-type algorithms the symbolic adjoint method can be applied either to the nonlinear system (NLS) itself or to the embedded linear (Newton) system (LS).

$\rightarrow$ N. Safiran, J. Lotz, J. Hüser

We consider the one dimensional nonlinear differential equation

$$\nabla^2(z \cdot u^*) + u^* \cdot \nabla(z \cdot u^*) = 0$$

on $\Omega = (0,1)$ with parameter $z(x)$ and for boundary conditions $u^* = 10,\ z = 1$ at $x = 0$ and $u^* = 20,\ z = 1$ at $x = 1$.
For given measurements $u^m(x)$ we aim to solve the parameter fitting problem

$$z^* = \arg\min_{z \in \mathbf{R}} \|u(x,z) - u^m(x)\|_2^2.$$

Equidistant central finite difference discretization yields the NLS

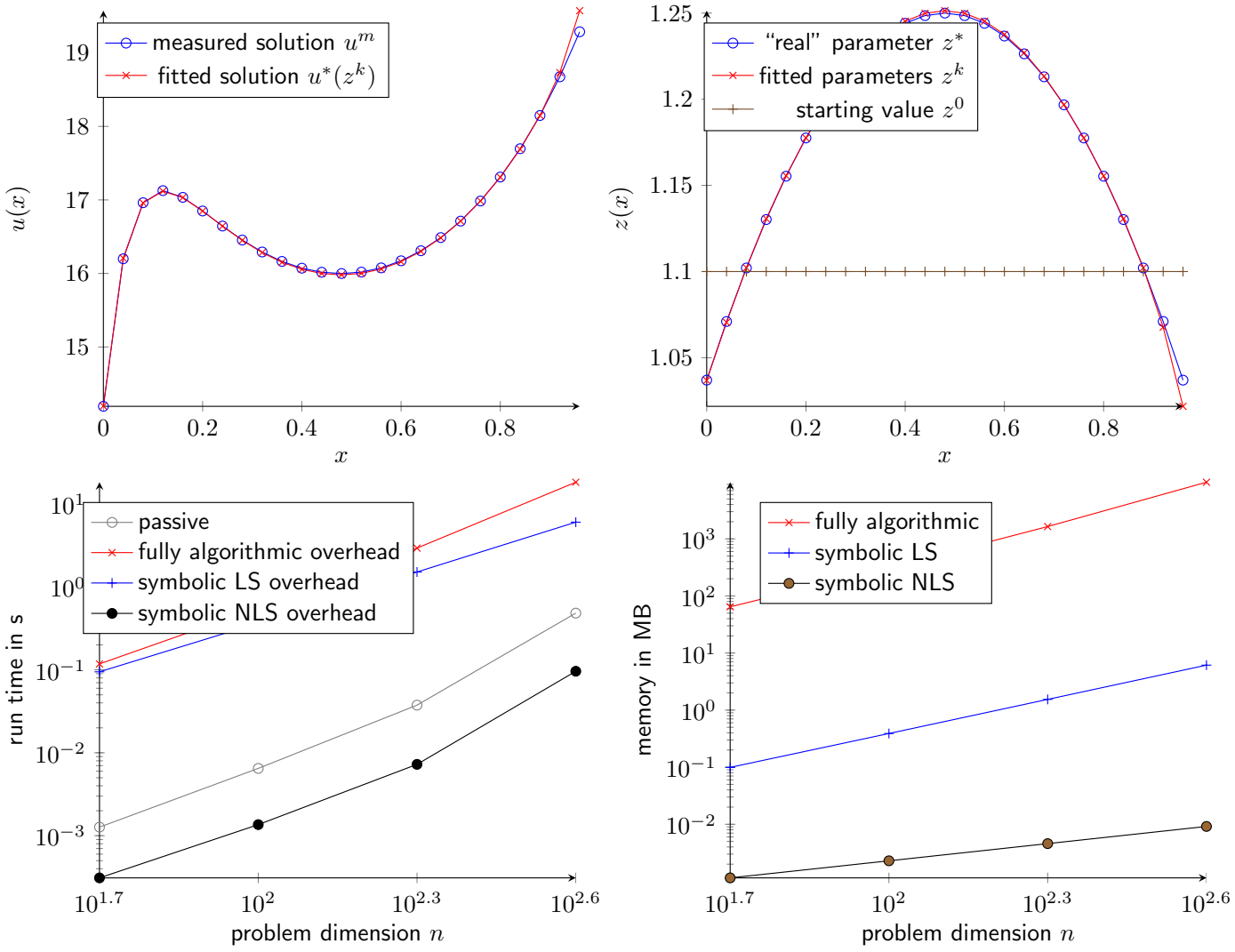
$$\mathbf{r}(\boldsymbol{u}, \mathbf{z}) = 0, \quad \boldsymbol{u} \in \mathbb{R}^n,\ \mathbf{z} \in \mathbb{R}^n\ ,$$

to solved by Newton's method yielding in the j-th iteration the LS

$$\frac{\partial \mathbf{r}}{\partial \mathbf{u}}(\mathbf{u}^j) \cdot \mathbf{s} = -\mathbf{r}(\boldsymbol{u}^j)\ .$$

The vector $\boldsymbol{u}^j$ is updated with the Newton step $\boldsymbol{u}^{j+1} = \boldsymbol{u}^j + \mathbf{s}$ for $j = 1, \ldots, \nu$.

Results [16]


measured solution u^m
fitted solution $u^*(z^k)$
$u(x)$
x
"real" parameter z^*
fitted parameters z^k
starting value z^0
$z(x)$
x
passive
fully algorithmic overhead
symbolic LS overhead
symbolic NLS overhead
run time in s
problem dimension n
fully algorithmic
symbolic LS
symbolic NLS
memory in MB
problem dimension n

Mind the Gap

NAG AD Library

Collaboration with the Numerical Algorithms Group Ltd.[1] dates back to 2000. Recent efforts aim for adjoint version of NAG library routines.
For example, nag_zero_cont_func_brent locates a simple zero x of a continuous function f in a given interval [a,b] using Brent's method [2]. The adjoint version enables computation of sensitivities of the solution x wrt. all relevant input parameters (potentially passed through comm).

```
void
nag_zero_cont_func_brent_dco_a1s (
  dco::a1s::type a, // in: lower bound
  dco::a1s::type b, // in: upper bound
  dco::a1s::type eps, // in: termination tolerance on x
  dco::a1s::type eta, // in: acceptance tolerance for vanishing f(x)
  dco::a1s::type (*f)(dco::a1s::type x, Nag_Comm_dco_a1s *comm), // in: f
  dco::a1s::type *x, // out: x
  Nag_Comm_dco_a1s *comm, // inout: parameters
  NagError *fail); // out: error code
```

→ V. Mosenkis, K. Leppkes

[1] www.nag.co.uk

Outline

Further Ongoing Projects

- Claix: Cluster Aix-la-Chapelle
- ADE: Aachen Dynamic Optimization Environment
- SCoRPiO: Significance-Based Computing for Reliability and Power Optimization
- ICON: (Icosahedral Non-hydrostatic) General Circulation Model
- TELEMAC-MASCARET: Free-Surface Flow Solver Library
- JURASSIC2: Juelich Rapid Spectral Simulation Code Version 2
- Medical Image Reconstruction

Further Issues

- Adjoint Shared Memory Parallelism (M. Förster)
- Adjoint Distributed Memory Parallelism (M. Schanen)
- Deterministic Global Optimization by McCormick Relaxations (M. Beckers)
- Robust Optimization by Moments Method (M. Beckers)
- Elimination techniques for Jacobians (V. Mosenkis)
- Optimal Call Tree Reversal (J. Lotz)

Conclusion

Watch out for ...

- API
- Robustness
- Efficiency
- Sustainability
- User Education
- Service/Support

... when planning your adjoint AD strategy!

Bibliography I

C. Bischof, M. Bücker, P. Hovland, U. Naumann, and J. Utke, editors.
Advances in Automatic Differentiation, volume 64 of *Lecture Notes in Computational Science and Engineering*.
Springer, Berlin, 2008.

R. Brent.
An algorithm with guaranteed convergence for finding a zero of a function.
Comput. J., 14(4):422–425, 1971.

B. Christianson.
Reverse accumulation and attractive fixed points.
Optimization Methods and Software, 3(4):311–326, 1994.

M. Giles.
Collected matrix derivative results for forward and reverse mode algorithmic differentiation.
In C. Bischof, M. Bücker, P. Hovland, U. Naumann, and J. Utke, editors, *Advances in Automatic Differentiation*, pages 35–44. Springer, 2008.

Bibliography II

F. Gremse, A. Höfter, L. Razik, F. Kießling, and U. Naumann.
GPU-accelerated adjoint algorithmic differentiation.
Submitted.

A. Griewank and C. Faure.
Reduced functions, gradients and Hessians from fixed-point iterations for state equations.
Numerical Algorithms, 30(2):113–139, 2002.

A. Griewank and A. Walther.
Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation.
ACM Transactions on Mathematical Software, 26(1):19–45, mar 2000.

G. Gundersen and T. Steihaug.
Sparsity in higher order methods for unconstrained optimization.
Optimization Methods and Software, 27(2):275–294, 2012.

Bibliography III

L. Hascoët, S. Fidanova, and C. Held.
Adjoining independent computations.
In G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors, *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Computer and Information Science, chapter 35, pages 299–304. Springer, New York, NY, 2002.

L. Hascoët and V. Pascual.
The Tapenade automatic differentiation tool: Principles, model, and specification.
ACM Transactions on Mathematical Software, 39(3):20:1–20:43, 2013.

R. J. Hogan.
Fast reverse-mode automatic differentiation using expression templates in C++.
ACM Transactions on Mathematical Software, 40(4):26:1–26:24, jun 2014.

Bibliography IV

S. Li and L. Petzold.
Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement.
Journal of Computational Physics, 198(1):310 – 325, 2004.

J. Lotz, U. Naumann, M. Sagebaum, and M. Schanen.
Discrete adjoints of PETSc through dco/c++ and adjoint MPI.
In *Euro-Par 2013 Parallel Processing*, pages 497–507. Springer, 2013.

U. Naumann.
Call Tree Reversal is NP-complete.
In *[1]*, pages 13–22. Springer, 2008.

U. Naumann and J. du Toit.
Adjoint algorithmic differentiation tool support for typical numerical patterns in computational finance.
Technical Report AIB2014-13, RWTH Aachen University, 2014.
Submitted.

Bibliography V

U. Naumann, J. Lotz, K. Leppkes, and M. Towara.
Algorithmic differentiation of numerical methods: Tangent and adjoint solvers for parameterized systems of nonlinear equations.
ACM Trans. Math. Soft., 2015.
To appear.

C. Othmer.
A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows.
International Journal for Numerical Methods in Fluids, 58(8):861–877, 2008.

S. V. Patankar and D.B. Spalding.
A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows.
Int. J. of Heat and Mass Transfer, 15(10):1787–1806, 1972.

Bibliography VI

L. Petzold, S. Li, Y. Cao, and R. Serban.
Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution.
SIAM J. Sci. Comput, 149(24):1076–1089, 2003.

M. Schanen, U. Naumann, L. Hascoët, and J. Utke.
Interpretative adjoints for numerical simulation codes using MPI.
Procedia Computer Science (Proceedings of ICCS 2010), pages 1825 – 1833, 2010.

M. Towara and U. Naumann.
A discrete adjoint model for OpenFOAM.
Procedia Computer Science (Proceedings of ICCS 2013), pages 429–438, 2013.

Bibliography VII

M. Towara and U. Naumann.
MPI-parallel discrete adjoint OpenFOAM.
2015.
To appear in Proceedings of ICCS 2015.