UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Four Ways (and Even More) to Compute Adjoints

Andrea Walther
Institut für Mathematik
Universität Paderborn

Woudschoten Conference 2012

October 3–5, 2012

Outline

Derivatives

- Optimisation:

  unconstrained: $\min f(x)$

  constrained: $\min f(x), \quad c(x) = 0\,, \;\; h(x) \leq 0$

- Solution of systems of nonlinear equations

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n$$

- Simulation of complicated systems
  - description of the system
  - integration of differential equations
- Sensitivity analysis
- Real time control

A (quite long) History of Differentiation

- Nasir ad-Din at-Tusi (1201-1274 Iran and Irak) examined positive solvability of
  $$c = f(x) = x^2(a - x)$$
  by maximizing $f(x)$ using derivative $f'(x)$.

A (quite long) History of Differentiation

- Nasir ad-Din at-Tusi (1201-1274 Iran and Irak) examined positive solvability of
  $$c = f(x) = x^2(a - x)$$
  by maximizing $f(x)$ using derivative $f'(x)$.

- Power series for trigonometric functions and inverses as well as their derivatives known in Kerala in late 1300's.

A (quite long) History of Differentiation

- Nasir ad-Din at-Tusi (1201-1274 Iran and Irak) examined positive solvability of
$$c = f(x) = x^2(a - x)$$
by maximizing $f(x)$ using derivative $f'(x)$.

- Power series for trigonometric functions and inverses as well as their derivatives known in Kerala in late 1300's.

- Hence: Definitely not just the Newton versus Leibniz controversy in 1666-1717

More recent statements

"Optimization problems, where the objective function is relatively expensive to compute and derivatives are not available, arise, for example, in engineering design, where the objective function evaluation is a simulation package treated as a black box."

Andrew Conn et al. 2008

More recent statements

"Optimization problems, where the objective function is relatively expensive to compute and derivatives are not available, arise, for example, in engineering design, where the objective function evaluation is a simulation package treated as a black box."

Andrew Conn et al. 2008

"Derivatives should not be regulated, they should be outlawed."

Badgerman@DemocraticUnderground on Obama proposal 2009.

More recent statements

“Optimization problems, where the objective function is relatively expensive to compute and derivatives are not available, arise, for example, in engineering design, where the objective function evaluation is a simulation package treated as a black box.”

Andrew Conn et al. 2008

"Derivatives should not be regulated, they should be outlawed."

Badgerman@DemocraticUnderground on Obama proposal 2009.

"There is a common misconception that calculating a function of n variables and its gradient is about $\mathbf{n+1}$ **times as expensive** as just calculating the function. This will only be true if the gradient is evaluated by differencing function values ... If care is taken in handling **quantities, which are common** to the function and its derivatives, **the ratio is usually 1.5**, not $n+1$; we have rarely seen a case where the ratio exceeds 2 "

Phil Wolfe, Transactions on Mathematical Software, 1984

Computing Derivatives

Given:

Description of functional relation as

- formula $y = F(x)$ ➔ explicit expression $y' = F'(x)$
- computer program ➔ ?

Computing Derivatives

Given:

Description of functional relation as

- formula $y = F(x)$ ➞ explicit expression $y' = F'(x)$
- computer program ➞ ?

Task:

Computation of derivatives taking

- requirements on exactness
- computational effort

into account

Finite Differences

Idea: Taylor-expansion, $f : \mathbb{R} \to \mathbb{R}$ smooth then

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + h^2 f''(x)/2 + h^3 f'''(x)/6 + \ldots \\ \Rightarrow \quad f(x+h) &\approx f(x) + hf'(x) \\ \Rightarrow \quad Df(x) &= \frac{f(x+h) - f(x)}{h} \end{aligned}$$

Finite Differences

Idea: Taylor-expansion, $f : \mathbb{R} \to \mathbb{R}$ smooth then

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + h^2 f''(x)/2 + h^3 f'''(x)/6 + \dots \\ \Rightarrow \quad f(x+h) &\approx f(x) + hf'(x) \\ \Rightarrow \quad Df(x) &= \frac{f(x+h) - f(x)}{h} \end{aligned}$$

- simple derivative calculation (only function evaluations!)
- inexact derivatives
- computation cost often too high

  $F : \mathbb{R}^n \to \mathbb{R} \quad \Rightarrow \quad \mathrm{OPS}(\nabla F(x)) \sim (n+1)\mathrm{OPS}(F(x))$

Symbolic/Analytic Differentiation

- exact derivatives
  - $$f(x) = \exp(\sin(x^2)) \Rightarrow$$
    $$f'(x) = \exp(\sin(x^2)) * \cos(x^2) * 2x$$

Symbolic/Analytic Differentiation

- exact derivatives
  - $f(x) \;=\; \exp(\sin(x^2)) \Rightarrow$

    $f'(x) \;=\; \exp(\sin(x^2)) \;*\; \cos(x^2) \;*\; 2x$
  - $\min J(x,u) \quad$ such that $\quad x' = f(x,u) \;+\;$ IC

    reduced formulation: $J(x,u) \rightarrow \widehat{J}(u)$

    $\widehat{J}'(u)$ based on symbolic adjoint $\lambda' = -f_x(x,u)^\top \lambda \;+\;$ TC

Symbolic/Analytic Differentiation

- exact derivatives
  - $f(x) \; = \; \exp(\sin(x^2)) \Rightarrow$

    $f'(x) \; = \; \exp(\sin(x^2)) \, * \, \cos(x^2) \, * \, 2x$
  - $\min J(x,u)$ such that $x' = f(x,u) \; + \;$ IC

    reduced formulation: $J(x,u) \rightarrow \widehat{J}(u)$

    $\widehat{J}'(u)$ based on symbolic adjoint $\lambda' = -f_x(x,u)^\top \lambda \; + \;$ TC
- cost (common subexpression, implementation)
- legacy code with large number of lines $\Rightarrow$
  closed form expression not available
- consistent derivative information?!

Example (Hager)

$$\min \frac{1}{2}\int_0^1 2x(t)^2 + u(t)^2 dt \quad \text{s.d.} \quad x' = \frac{1}{2}x(t) + u(t), \quad x(0) = 1$$

Example (Hager)

$$\min \frac{1}{2}\int_0^1 2x(t)^2 + u(t)^2 dt \quad \text{s.d.} \quad x' = \frac{1}{2}x(t) + u(t), \quad x(0) = 1$$

$$\lambda(t) = \frac{e^{3-t} - 2e^{2t}}{e^{t/2}(2+e^3)}$$

$$\lambda(1) = 0$$

Example (Hager)

$$\min \frac{1}{2}\int_0^1 2x(t)^2 + u(t)^2 dt \quad \text{s.d.} \quad x' = \frac{1}{2}x(t) + u(t), \quad x(0) = 1$$

$$\lambda(t) = \frac{e^{3-t} - 2e^{2t}}{e^{t/2}(2+e^3)}$$
$$\lambda(1) = 0$$

$$x^*(t) = (2e^{3t} + e^3)/(e^{3t/2}(2+e^3))$$
$$u^*(t) = 2(e^{3t} - e^3)/(e^{3t/2}(2+e^3))$$

Example (Hager)

$$\min \tfrac{1}{2} \int_0^1 2x(t)^2 + u(t)^2 dt \quad \text{s.d.} \quad x' = \tfrac{1}{2}x(t) + u(t), \quad x(0) = 1$$

$$\lambda(t) = \frac{e^{3-t} - 2e^{2t}}{e^{t/2}(2+e^3)}$$
$$\lambda(1) = 0$$

$$x^*(t) = (2e^{3t} + e^3)/(e^{3t/2}(2+e^3))$$
$$u^*(t) = 2(e^{3t} - e^3)/(e^{3t/2}(2+e^3))$$

$$\min \tfrac{h}{2} \sum_{i=0}^{N} 2x_{i+1/2}^2 + u_{i+1/2}^2$$
$$x_{i+1/2} = x_i + \tfrac{h}{2}(x_i + u_i)$$
$$x_{i+1} = x_i + \tfrac{h}{2}(\tfrac{1}{2}x_{i+1/2} + u_{i+1/2})$$

Example (Hager)

$$\min \tfrac{1}{2} \int_0^1 2x(t)^2 + u(t)^2 dt \quad \text{s.d.} \quad x' = \tfrac{1}{2}x(t) + u(t), \quad x(0) = 1$$

$$\lambda(t) = \frac{e^{3-t} - 2e^{2t}}{e^{t/2}(2+e^3)} \qquad \lambda(1) = 0$$

$\longrightarrow$

$$x^*(t) = (2e^{3t} + e^3)/(e^{3t/2}(2+e^3))$$
$$u^*(t) = 2(e^{3t} - e^3)/(e^{3t/2}(2+e^3))$$

$$\min \tfrac{h}{2} \sum_{i=0}^{N} 2x_{i+1/2}^2 + u_{i+1/2}^2$$
$$x_{i+1/2} = x_i + \tfrac{h}{2}(x_i + u_i)$$
$$x_{i+1} = x_i + \tfrac{h}{2}(\tfrac{1}{2}x_{i+1/2} + u_{i+1/2})$$

$\longrightarrow$

$$x_i = 1, \; x_{i+1/2} = 0$$
$$u_i = -\frac{4+h}{2h}x_i, \; u_{i+1/2} = 0$$

```
  read_input_file(argv[1], &code_control);

  code_control.timestep_type = 0; // calculate timestep size like TAU

  // read in CFD mesh
  read_cfd_mesh(code_control.CFDmesh_name, &gridbase);
  grid[0] = gridbase;

  // remove mesh corner points arizing more than once . . .
  // e.g. for block structured area and at interface between
  // block structured and unstructured area
  remove_double_points( &gridbase, grid);

  // write out mesh in tecplot format
  write_pointdata( name, &(grid[0]));

  // calculate metric of finest grid level
/*     grid[0].xp[ii][1] += 0.00000001; */
  calc_metric(&(grid[0]), &code_control);
  puts("calc_metric ready");

  // create coarse meshes for multigrid, calculate their metric
  // and initialze forcing functions to zero
  for (i = 1; i < code_control.nlevels; i++)
  { create_coarse_mesh(&(grid[i-1]), &(grid[i]));
    init2zero(&(grid[i]), grid[i].force);
  }
  puts("create_coarse_mesh ready");

  // initialize flow field on all grid levels to free stream
  // quantities
  for (i = 0; i < code_control.nlevels; i++)
    init_field(&(grid[i]), &code_control);
  puts("init_field ready");

  // if selected read restart file
  if (code_control.restart == 1)
    read_restart( "restart", grid, &code_control,
                  &first_residual, &first_step);

  // calculate primitive variables for all grid levels and
  // initialize states at the boundary
  for (i = 0; i < code_control.nlevels; i++)
  { cons2prim(&(grid[i]), &code_control);
    init_bdry_states(&(grid[i]));
  }

  // open file for writing convergence history
  conv = fopen("conv.dat", "w");
  fprintf(conv,  "title = convergence\n");
  fprintf(conv,  "variables = iter, l2res, lift, drag\n");


  level = 0;
  printf("will perform %d steps\n",code_control.nsteps[level]);

  // starting time of computation
  t1 = time(&t1);

  double lift, drag;

  // loop over all multigrid cycles
```



```
  for (it = 0; it < code_control.nsteps[level]; it++)
  { double residual;
    lift = 0.0;
    drag = 0.0;

    // calculate actual weight of gradient needed for reconstruction
    if (sum_it+first_step <= code_control.start_2nd_order)
      weight = 0.0;
    else if (sum_it+first_step < code_control.full_2nd_order)
      weight = (double ) (sum_it+first_step - code_control.start_2nd_order) /
                         (code_control.full_2nd_order - code_control.start_2nd_o
rder);
    else
      weight = 1.0;

    // perform a multigrid cycle on current level
    mg_cycle(grid+level, &code_control, weight, &residual);

    // if current level is finest level, calculate boundary forces
    // (lift and drag)
    if (level == 0)
      calc_forces(grid, &code_control, &lift, &drag);

    // set first l2-residual for normalization, if current cycle is
    // the very first of the computation.
    if ((sum_it + first_step) == 0)
      first_residual = (fabs(residual) > 1.0e-10) ? residual: 1.0;

    // print out convergence information to file and standard output
    printf("IT = %d %20.10e %20.10e %20.10e %4.2f\n",
      sum_it, residual / first_residual, lift, drag, weight);
    fprintf(conv, "%d %20.10e %20.10e %20.10e\n",
      sum_it+first_step, residual / first_residual, lift, drag);
    sum_it++;
  }

  // final time of computation
  t2 = time(&t2);

  // print out time needed for the time loop
  printf ("Zeit : %f\n", difftime(t2, t1));
  last_step = first_step + code_control.nsteps[0] ;

  fclose(conv);

  // map solution from cell centers to vertices
  center2point(grid);

  // write out field solution
  write_eulerdata( "euler.dat", grid, &code_control);

  // write out solution on walls
  write_surf( "euler-surf.dat", grid, &code_control);

  // write restart file
  write_restart( "restart", grid, &code_control,
                 first_residual, last_step);

  return 0;
}
```

Algorithmic Differentiation (AD)

Main Products:

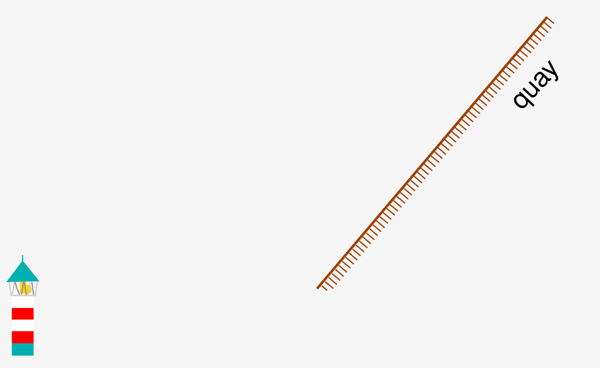
- Quantitative dependence information (local):
  - Weighted and directed partial derivatives
  - Error and condition number estimates . . .
  - Lipschitz constants, interval enclosures . . .
  - Eigenvalues, Newton steps . . .

Algorithmic Differentiation (AD)

Main Products:

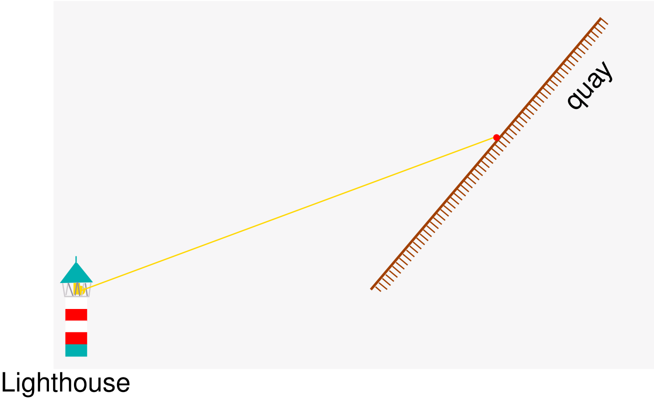
- Quantitative dependence information (local):
  - Weighted and directed partial derivatives
  - Error and condition number estimates . . .
  - Lipschitz constants, interval enclosures . . .
  - Eigenvalues, Newton steps . . .
- Qualitative dependence information (regional):
  - Sparsity structures, degrees of polynomials
  - Ranks, eigenvalue multiplicities . . .

The “Hello-World”-Example of AD

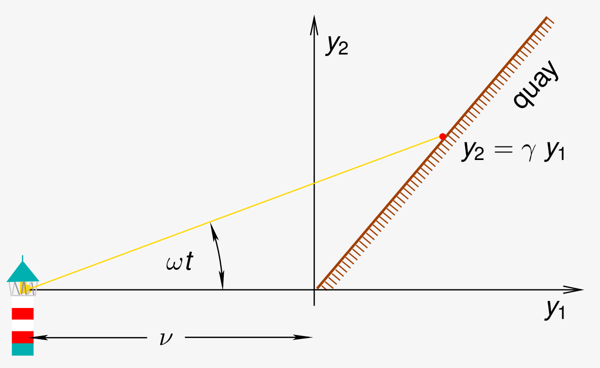


Lighthouse

The “Hello-World”-Example of AD

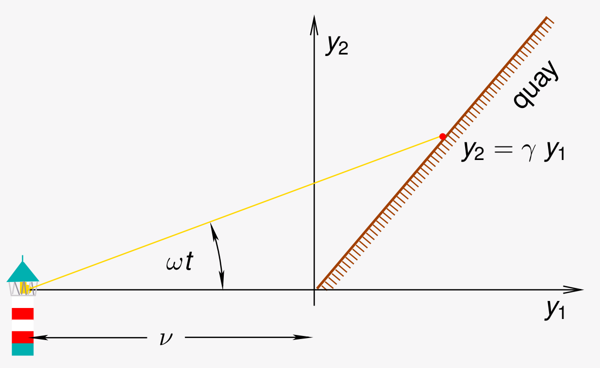


The “Hello-World”-Example of AD



Lighthouse

The “Hello-World”-Example of AD



Lighthouse

$$y_1 = \frac{\nu \tan(\omega\, t)}{\gamma - \tan(\omega\, t)} \quad \text{and} \quad y_2 = \frac{\gamma\, \nu \tan(\omega\, t)}{\gamma - \tan(\omega\, t)}$$

Evaluation Procedure (Lighthouse)

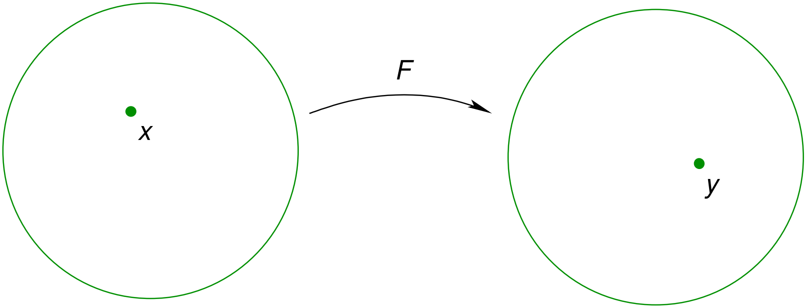
$$y_1 = \frac{\nu \tan(\omega t)}{\gamma - \tan(\omega t)}$$

$$y_2 = \frac{\gamma \nu \tan(\omega t)}{\gamma - \tan(\omega t)}$$

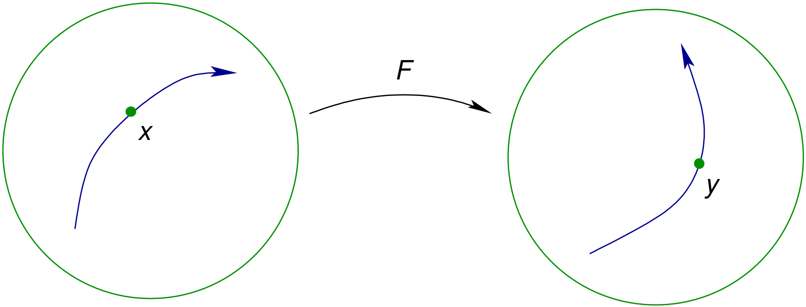
$\Longrightarrow$

$$\begin{array}{lll}
v_{-3} & = x_1 = \nu & \\
v_{-2} & = x_2 = \gamma & \\
v_{-1} & = x_3 = \omega & \\
v_0 & = x_4 = t & \\
\hline
v_1 & = v_{-1} * v_0 & \equiv \varphi_1(v_{-1}, v_0) \\
v_2 & = \tan(v_1) & \equiv \varphi_2(v_1) \\
v_3 & = v_{-2} - v_2 & \equiv \varphi_3(v_{-2}, v_2) \\
v_4 & = v_{-3} * v_2 & \equiv \varphi_4(v_{-3}, v_2) \\
v_5 & = v_4 / v_3 & \equiv \varphi_5(v_4, v_3) \\
v_6 & = v_5 * v_{-2} & \equiv \varphi_6(v_5, v_{-2}) \\
\hline
y_1 & = v_5 & \\
y_2 & = v_6 &
\end{array}$$

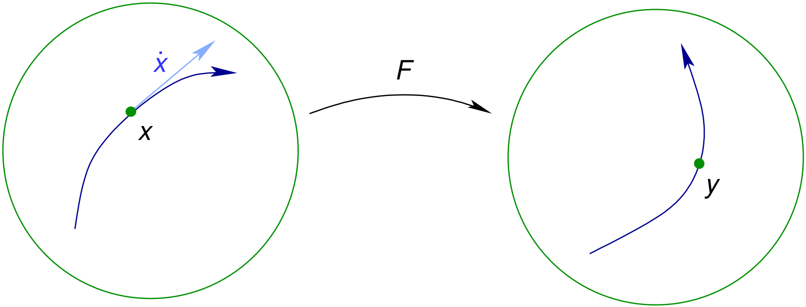
Forward mode AD = Tangents/Sensitivities



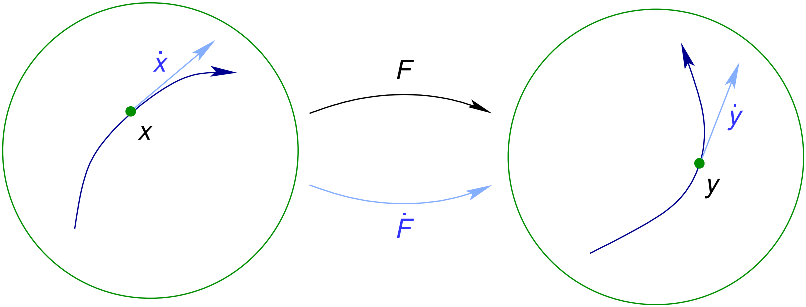
Forward mode AD = Tangents/Sensitivities



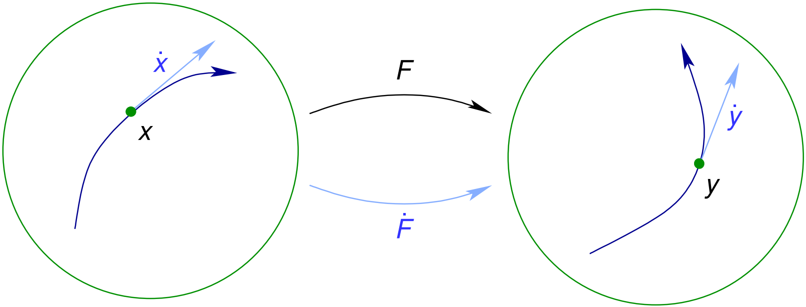
Forward mode AD = Tangents/Sensitivities



Forward mode AD = Tangents/Sensitivities



Forward mode AD = Tangents/Sensitivities



$$\dot{y}(t) \;=\; \frac{\partial}{\partial t} F(x(t)) \;=\; F'(x(t))\,\dot{x}(t) \equiv \; \dot{F}(x, \dot{x})$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl}
v_{-3} & = & x_1 = \nu \\
v_{-2} & = & x_2 = \gamma \\
v_{-1} & = & x_3 = \omega \\
v_0 & = & x_4 = t \\
\hline
v_1 & = & v_{-1} * v_0 \\
v_2 & = & \tan(v_1) \\
v_3 & = & v_{-2} - v_2 \\
v_4 & = & v_{-3} * v_2 \\
v_5 & = & v_4 / v_3 \\
v_6 & = & v_5 * v_{-2} \\
\hline
y_1 & = & v_5 \\
y_2 & = & v_6
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & & & \\
v_2 & = & \tan(v_1) & & & \\
v_3 & = & v_{-2} - v_2 & & & \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lll|lll}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & & & \\
v_3 & = & v_{-2} - v_2 & & & \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & & & \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lll|lll}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{rcl|rcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

v_{-3}	$=$	$x_1 = \nu$	$\dot{v}_{-3}$	$=$	$\dot{x}_1$
v_{-2}	$=$	$x_2 = \gamma$	$\dot{v}_{-2}$	$=$	$\dot{x}_2$
v_{-1}	$=$	$x_3 = \omega$	$\dot{v}_{-1}$	$=$	$\dot{x}_3$
v_0	$=$	$x_4 = t$	$\dot{v}_0$	$=$	$\dot{x}_4$
v_1	$=$	$v_{-1} * v_0$	$\dot{v}_1$	$=$	$\dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0$
v_2	$=$	$\tan(v_1)$	$\dot{v}_2$	$=$	$\dot{v}_1 / \cos(v_1)^2$
v_3	$=$	$v_{-2} - v_2$	$\dot{v}_3$	$=$	$\dot{v}_{-2} - \dot{v}_2$
v_4	$=$	$v_{-3} * v_2$	$\dot{v}_4$	$=$	$\dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2$
v_5	$=$	v_4 / v_3	$\dot{v}_5$	$=$	$(\dot{v}_4 - \dot{v}_3 * v_5) * (1/v_3)$
v_6	$=$	$v_5 * v_{-2}$			
y_1	$=$	v_5			
y_2	$=$	v_6			

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & \dot{v}_5 & = & (\dot{v}_4 - \dot{v}_3 * v_5) * (1/v_3) \\
v_6 & = & v_5 * v_{-2} & \dot{v}_6 & = & \dot{v}_5 * v_{-2} + v_5 * \dot{v}_{-2} \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & \dot{v}_5 & = & (\dot{v}_4 - \dot{v}_3 * v_5) * (1 / v_3) \\
v_6 & = & v_5 * v_{-2} & \dot{v}_6 & = & \dot{v}_5 * v_{-2} + v_5 * \dot{v}_{-2} \\
\hline
y_1 & = & v_5 & \dot{y}_1 & = & \dot{v}_5 \\
y_2 & = & v_6 & \dot{y}_2 & = & \dot{v}_6
\end{array}
$$

... and the real code

```
void d1_f(double* x, double* d1_x, double* y, double* d1_y)
//$ad indep x d1_x
//$ad dep y d1_y
 {
   double v[2];                double d1_v[2];
   double w1_0 = 0;            double d1_w1_0 = 0;
   ...
   double w1_5 = 0;            double d1_w1_5 = 0;

   d1_w1_0 = d1_x[2];          w1_0 = x[2];
   d1_w1_1 = d1_x[3];          w1_1 = x[3];
   d1_w1_2 = w1_1*d1_w1_0 + w1_0*d1_w1_1;
   w1_2 = w1_0*w1_1;
   d1_w1_3 = 1/(cos(w1_2)*cos(w1_2)) * d1_w1_2;
   w1_3 = tan(w1_2);
   ...
```

... and the real code

```
void d1_f(double* x, double* d1_x, double* y, double* d1_y)
//$ad indep x d1_x
//$ad dep y d1_y
 {
   double v[2];                double d1_v[2];
   double w1_0 = 0;            double d1_w1_0 = 0;
   ...
   double w1_5 = 0;            double d1_w1_5 = 0;

   d1_w1_0 = d1_x[2];          w1_0 = x[2];
   d1_w1_1 = d1_x[3];          w1_1 = x[3];
   d1_w1_2 = w1_1*d1_w1_0 + w1_0*d1_w1_1;
   w1_2 = w1_0*w1_1;
   d1_w1_3 = 1/(cos(w1_2)*cos(w1_2)) * d1_w1_2;
   w1_3 = tan(w1_2);
   ...
```

... and the real code

```
void d1_f(double* x, double* d1_x, double* y, double* d1_y)
//$ad indep x d1_x
//$ad dep y d1_y
 {
   double v[2];                 double d1_v[2];
   double w1_0 = 0;             double d1_w1_0 = 0;
   ...
   double w1_5 = 0;             double d1_w1_5 = 0;

   d1_w1_0 = d1_x[2];           w1_0 = x[2];
   d1_w1_1 = d1_x[3];           w1_1 = x[3];
   d1_w1_2 = w1_1*d1_w1_0 + w1_0*d1_w1_1;
   w1_2 = w1_0*w1_1;
   d1_w1_3 = 1/(cos(w1_2)*cos(w1_2)) * d1_w1_2;
   w1_3 = tan(w1_2);
   ...
```

using dcc 1.0 (U. Naumann, RWTH Aachen)

Complexity (Forward Mode)

tang	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+3$	$3+3$	$2+2$
ADDS	0	$1+1$	$0+1$	$0+0$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

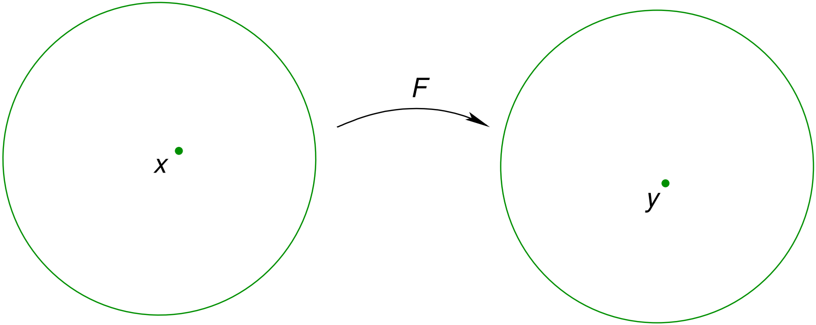
$$\text{OPS}(F'(x)\dot{x}) \quad \leq \quad c\,\text{OPS}(F(x))$$

with $c \in [2, 5/2]$ platform dependent

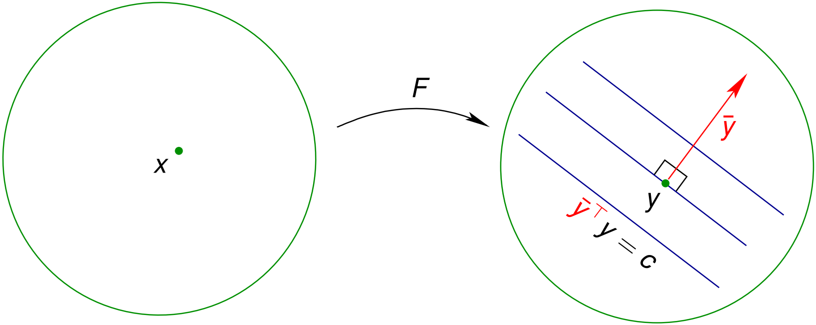
Relation to Continuous Formulation

- Discrete analogon to sensitivity equation
- Several theoretical results
  - Convergence order for ODE-based Optimization
    Walther [2008], Sandu [200*]
  - Convergence analysis for adaptive time stepping
    Eberhardt and Bischof [1999], Alexe and Sandu [2009]
  - Probably easy to extend to FEM discretization
- (Black Box) Application to FEM-based Simulation
  e.g. Bischof and Behr (SPP 1253), Bücker et al. [200*],
  Kowarz and Walther [2007], . . .

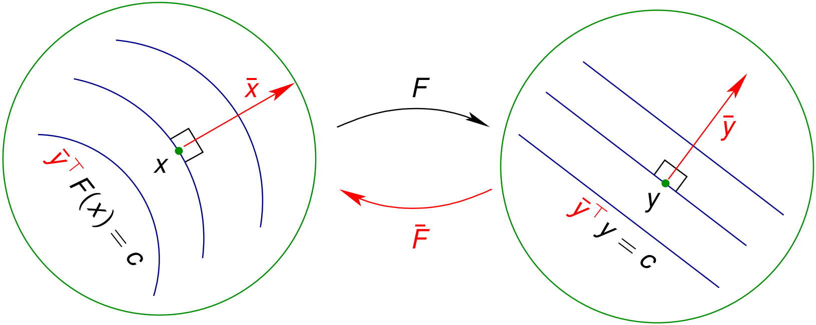
Reverse Mode AD = Discrete Adjoints



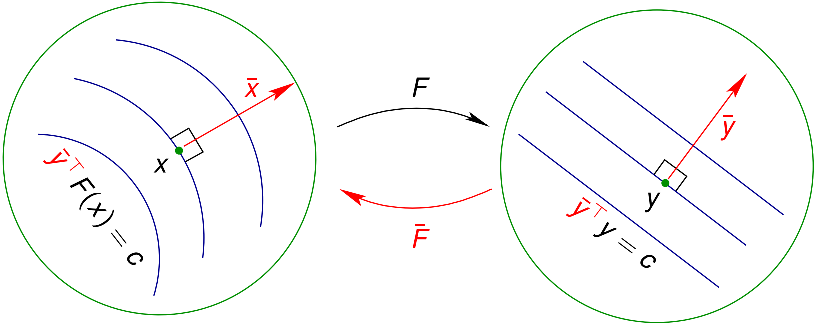
Reverse Mode AD = Discrete Adjoints



Reverse Mode AD = Discrete Adjoints



Reverse Mode AD = Discrete Adjoints



$$\bar{x} \equiv \bar{y}^\top F'(x) = \nabla_x \langle \bar{y}^\top F(x) \rangle \equiv \bar{F}(x, \bar{y})$$

Reverse Mode (Lighthouse)

$$
\begin{aligned}
&v_{-3} = x_1; \quad v_{-2} = x_2; \quad v_{-1} = x_3; \quad v_0 = x_4;\\
&v_1 = v_{-1} * v_0;\\
&v_2 = \tan(v_1);\\
&v_3 = v_{-2} - v_2;\\
&v_4 = v_{-3} * v_2;\\
&v_5 = v_4 / v_3;\\
&v_6 = v_5 * v_{-2};\\
&y_1 = v_5; \quad y_2 = v_6;\\
\hline
&\bar{v}_5 = \bar{y}_1; \quad \bar{v}_6 = \bar{y}_2;\\
&\bar{v}_5 \mathrel{+}= \bar{v}_6 * v_{-2}; \quad \bar{v}_{-2} \mathrel{+}= \bar{v}_6 * v_5;\\
&\bar{v}_4 \mathrel{+}= \bar{v}_5 / v_3; \quad \bar{v}_3 \mathrel{-}= \bar{v}_5 * v_5 / v_3;\\
&\bar{v}_{-3} \mathrel{+}= \bar{v}_4 * v_2; \quad \bar{v}_2 \mathrel{+}= \bar{v}_4 * v_{-3};\\
&\bar{v}_{-2} \mathrel{+}= \bar{v}_3; \; \bar{v}_2 \mathrel{-}= \bar{v}_3;\\
&\bar{v}_1 \mathrel{+}= \bar{v}_2 / \cos^2(v_1);\\
&\bar{v}_{-1} \mathrel{+}= \bar{v}_1 * v_0; \; \bar{v}_0 \mathrel{+}= \bar{v}_1 * v_{-1};\\
&\bar{x}_4 = \bar{v}_0; \quad \bar{x}_3 = \bar{v}_{-1}; \quad \bar{x}_2 = \bar{v}_{-2}; \quad \bar{x}_1 = \bar{v}_{-3};
\end{aligned}
$$

... and the real code generated by dcc 1.0

```
void b1_f(int& bmode_1, double* x, double* b1_x, double* y, double* b1_y)
//$ad indep x b1_x b1_y
//$ad dep y b1_x
 { double v[2];          double b1_v[2];
   double w1_0 = 0;      double b1_w1_0 = 0;      . . .
   double w1_5 = 0;      double b1_w1_5 = 0;
   int save_cs_c = 0;    save_cs_c = cs_c;
   if (bmode_1==1) { // augmented forward section
    cs[cs_c] = 0;          cs_c = cs_c+1;
    fds[fds_c] = v[0];     fds_c = fds_c+1;    v[0] = tan(x[2]*x[3]);
    . . .
    fds[fds_c] = y[1];     fds_c = fds_c+1;    y[1] = x[1]*y[0];
    while (cs_c>save_cs_c) {    // reverse section
       cs_c = cs_c-1;
       if (cs[cs_c]==0) {
          fds_c = fds_c-1;       y[1] = fds[fds_c];
          w1_0 = x[1];           w1_1 = y[0];      w1_2 = w1_0*w1_1;
          b1_w1_2 = b1_y[1];  b1_y[1] = 0; // adjoint assignment
          b1_w1_0 = w1_1*b1_w1_2;    b1_w1_1 = w1_0*b1_w1_2;
          b1_y[0] = b1_y[0]+b1_w1_1;   b1_x[1] = b1_x[1]+b1_w1_0;      . . .
```

Complexity (Reverse Mode)

grad	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+6$	$3+8$	$2+5$
ADDS	0	$1+2$	$0+2$	$0+1$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

$$\begin{aligned} \text{OPS}(\bar{y}^\top F'(x)) &\leq c\,\text{OPS}(F(x)) \\ \text{MEM}(\bar{y}^\top F'(x)) &\sim \text{OPS}(F(x)) \end{aligned}$$

with $c \in [3,4]$ platform dependent

Complexity (Reverse Mode)

grad	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+6$	$3+8$	$2+5$
ADDS	0	$1+2$	$0+2$	$0+1$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

$$\Rightarrow \quad \begin{array}{lcl} \text{OPS}(\bar{y}^\top F'(x)) & \leq & c\,\text{OPS}(F(x)) \\ \text{MEM}(\bar{y}^\top F'(x)) & \sim & \text{OPS}(F(x)) \end{array}$$

with $c \in [3, 4]$ platform dependent

Remarks:

- Cost for gradient calculation independent of n
- Memory requirement may cause problem! $\Rightarrow$ Checkpointing

Relation to Continuous Formulation

- Discrete analogon to adjoint equation
- Consistent discretization ?!
  Example: Explicit Euler scheme
- Several theoretical results
  - Convergence order for ODE-based Optimization
    Walther [2008], Sandu [200*]
  - Convergence analysis for adaptive time stepping
    Alexe and Sandu [2009]
  - Convergence order for FEM-based Discretization
- (Black Box !!) Application to FEM-based Simulation
  usually inappropriate due to memory requirement!!

Conclusions: Basic AD

- Evaluation of derivatives with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- Forward mode: $\text{OPS}(F'(x)\dot{x}) \leq c\,\text{OPS}(F), \quad c \in [2, 5/2]$
  Reverse mode: $\text{OPS}(\bar{y}^\top F'(x)) \leq c\,\text{OPS}(F), \quad c \in [3, 4]$
  $\text{MEM}(\bar{y}^\top F'(x)) \sim \text{OPS}(F),$

⟹ Gradients are cheap $\sim$ Function Costs!!

Conclusions: Basic AD

- Evaluation of derivatives with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- Forward mode: $\mathrm{OPS}(F'(x)\dot{x}) \leq c\,\mathrm{OPS}(F), \quad c \in [2, 5/2]$
  Reverse mode: $\mathrm{OPS}(\bar{y}^\top F'(x)) \leq c\,\mathrm{OPS}(F), \quad c \in [3, 4]$
  $\mathrm{MEM}(\bar{y}^\top F'(x)) \sim \mathrm{OPS}(F),$

⟹ Gradients are cheap $\sim$ Function Costs!!

- Combination: $\mathrm{OPS}(\bar{y}^\top F''(x)\dot{x}) \leq c\,\mathrm{OPS}(F), \quad c \in [7, 10]$
- Cost of higher derivatives grows quadratically in the degree
- Nondifferentiability only on meager set
- Full Jacobians/Hessians often not needed or sparse

Conclusions: Basic AD

- Evaluation of derivatives with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- Forward mode: $\mathrm{OPS}(F'(x)\dot{x}) \leq c\,\mathrm{OPS}(F), \quad c \in [2, 5/2]$
  Reverse mode: $\mathrm{OPS}(\bar{y}^\top F'(x)) \leq c\,\mathrm{OPS}(F), \quad c \in [3, 4]$
  $\mathrm{MEM}(\bar{y}^\top F'(x)) \sim \mathrm{OPS}(F),$

⟶ Gradients are cheap $\sim$ Function Costs!!

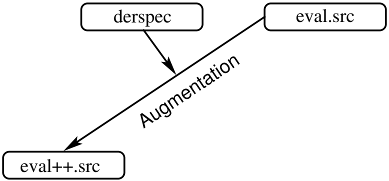
- Combination: $\mathrm{OPS}(\bar{y}^\top F''(x)\dot{x}) \leq c\,\mathrm{OPS}(F), \quad c \in [7, 10]$
- Cost of higher derivatives grows quadratically in the degree
- Nondifferentiability only on meager set
- Full Jacobians/Hessians often not needed or sparse

Questions: Structure Exploitation!!
Time-stepping, sparsity, fixed point iteration, ...

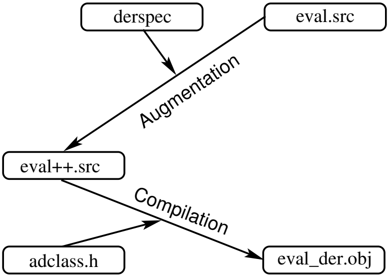
From Source Code to Derived Object Files

eval.src

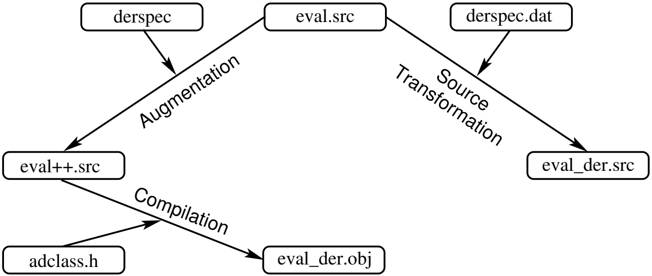
From Source Code to Derived Object Files



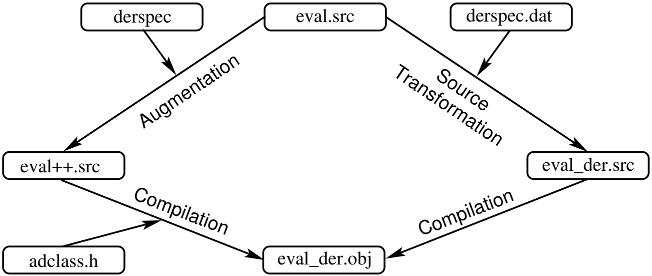
From Source Code to Derived Object Files



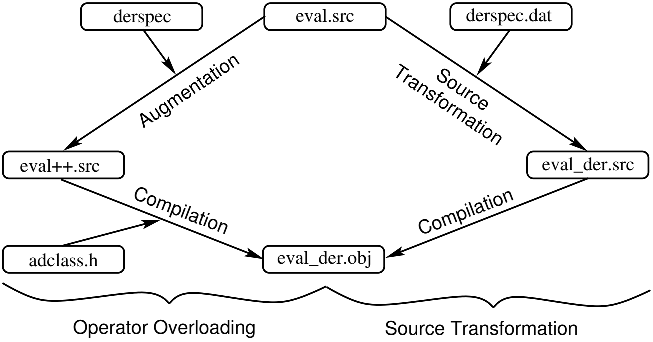
From Source Code to Derived Object Files



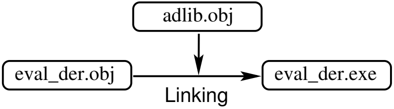
From Source Code to Derived Object Files



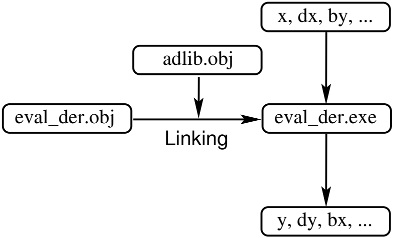
From Source Code to Derived Object Files



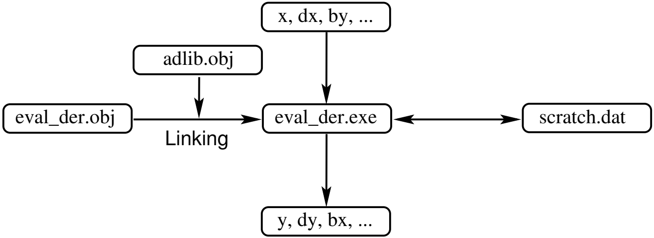
Linking and Executing Object Files



Linking and Executing Object Files



Linking and Executing Object Files



AD Tools

Fortran 77 (90): (mainly source transformation)

- Tapenade (INRIA, F)
- AD in the compiler (NAG, RWTH Aachen, Univ. Hertfordshire)
- ...

AD Tools

Fortran 77 (90): (mainly source transformation)

- Tapenade (INRIA, F)
- AD in the compiler (NAG, RWTH Aachen, Univ. Hertfordshire)
- ...

C/C++: (mainly operator overloading)

- ADOL-C (Univ. Paderborn)
- CppAD (Univ. Washington, USA)
- ...

AD Tools

Fortran 77 (90): (mainly source transformation)

- Tapenade (INRIA, F)
- AD in the compiler (NAG, RWTH Aachen, Univ. Hertfordshire)
- ...

C/C++: (mainly operator overloading)

- ADOL-C (Univ. Paderborn)
- CppAD (Univ. Washington, USA)
- ...

Matlab: Adimat, MAD, ...

AD Tools

Fortran 77 (90): (mainly source transformation)

- Tapenade (INRIA, F)
- AD in the compiler (NAG, RWTH Aachen, Univ. Hertfordshire)
- ...

C/C++: (mainly operator overloading)

- ADOL-C (Univ. Paderborn)
- CppAD (Univ. Washington, USA)
- ...

Matlab: Adimat, MAD, ...

see www.autodiff.org, [Griewank, Walther 2008], [Naumann 2012] for more tools and literature

Automatic Differentiation by Overloading in C++

- **ADOL-C version 2.3**
- available at COIN-OR since May 2009
- interface to ColPack (Purdue University) and Ipopt (COIN-OR)
- python wrapper available (Sebastian Walter, HUB)
- recent developments
  - improved computation of sparsity pattern for Hessians
  - handling of MPI-parallel/GPU-parallel codes
- future plans
  - generalized derivatives for nonsmooth functions
  - ...

Optimal Power Flow Problem

(Fabrice Zaoui, Laure Castaing, RTE France)

Task: Distribute power flow over given network

Difficulty: Unabservable areas due to

- lack of sensors
- error in data transmission
- ...

Approximate required data in unabservable areas ⟶

$$\begin{array}{ccc} \min f(x), & c(x) = 0 & h(x) \leq 0, \\ f : \mathbb{R}^n \to \mathbb{R}, & c : \mathbb{R}^n \to \mathbb{R}^m, & h : \mathbb{R}^n \to \mathbb{R}^p \end{array}$$

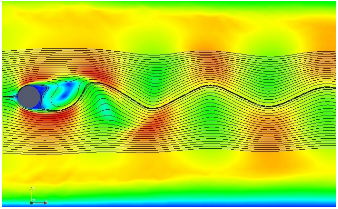
Optimal Power Flow (Discretizations)

n	m	p	$nnz(c)$	$nnz(h)$	$nnz(L)$	time (s)
5,986	2,415	1,575	21,065	6,300	21,068	11
17,958	7,245	11,123	63,179	31,692	64,668	55
29,930	12,075	20,671	105,301	57,084	108,278	129
53,874	21,735	39,767	189,529	107,868	195,478	412
101,762	41,055	77,959	358,025	209,436	369,916	1326

using ADOL-C + ColPack + interior point method

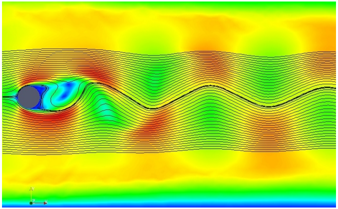
Real Time-dependent Problems

- Example:
  Transient flows
- Target: Minimize drag/turbulence



Real Time-dependent Problems

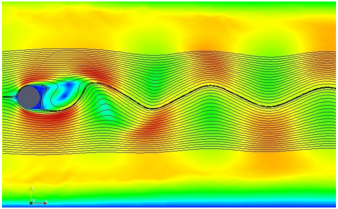
- Example:
  Transient flows
- Target: Minimize drag/turbulence



Approach:

- Adjoint of one time step only !! . . .

Real Time-dependent Problems



- Example:
  Transient flows
- Target: Minimize drag/turbulence

Approach:

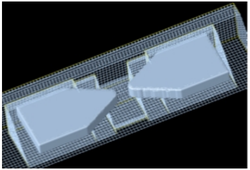
- Adjoint of one time step only !! . . .
- Checkpointing in all variations
  see talk tomorrow

Optical Nano-Structures

State of the art:
Nano-structures are used to confine light

Simple example: Bow-tie antenna

- metallic nano structure
- two triangles and gap
- Size: 100 nm ($< \lambda_{\text{light}}$!)
- intensity enhancement in gap

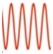


Possible Configurations

So far:

- different structure

- "simple" excitation



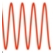
- pure metal
- extremely short dephasing

Possible Configurations

So far:

- different structure

- “simple” excitation



- pure metal
- extremely short dephasing

Now:

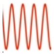
- fixed structure

Possible Configurations

So far:

- different structure

- “simple” excitation

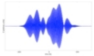


- pure metal
- extremely short dephasing

Now:

- fixed structure

- sophisticated excitation

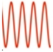


Possible Configurations

So far:

- different structure

- “simple” excitation

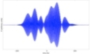


- pure metal
- extremely short dephasing

Now:

- fixed structure

- sophisticated excitation



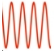
- add resonances in semiconductors

Possible Configurations

So far:

- different structure

- “simple” excitation

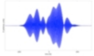


- pure metal
- extremely short dephasing

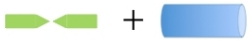
Now:

- fixed structure

- sophisticated excitation



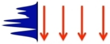
- add resonances in semiconductors
- longer dephasing



Nanooptics: Test Case

Cooperation with T. Meier, M. Reichelt, Dep. Physik, Uni Paderborn

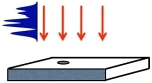
Generic configuration:

 $\longleftarrow$ adaptable light puls $E(t)$

Nanooptics: Test Case

Cooperation with T. Meier, M. Reichelt, Dep. Physik, Uni Paderborn

Generic configuration:



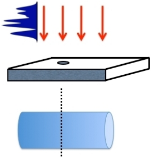
$\longleftarrow$ adaptable light puls $E(t)$

$\longleftarrow$ metal aperture with air hole

Nanooptics: Test Case

Cooperation with T. Meier, M. Reichelt, Dep. Physik, Uni Paderborn

Generic configuration:



$\longleftarrow$ adaptable light puls $E(t)$

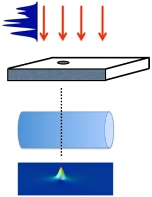
$\longleftarrow$ metal aperture with air hole

$\longleftarrow$ quantum wire

Nanooptics: Test Case

Cooperation with T. Meier, M. Reichelt, Dep. Physik, Uni Paderborn

Generic configuration:



⟵ adaptable light puls $E(t)$

⟵ metal aperture with air hole

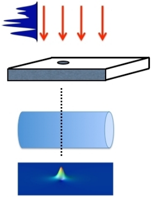
⟵ quantum wire

⟵ electron distribution?

Nanooptics: Test Case

Cooperation with T. Meier, M. Reichelt, Dep. Physik, Uni Paderborn

Generic configuration:



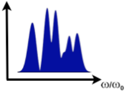
$\longleftarrow$ adaptable light puls $E(t)$

$\longleftarrow$ metal aperture with air hole

$\longleftarrow$ quantum wire

$\longleftarrow$ electron distribution?

Light puls:

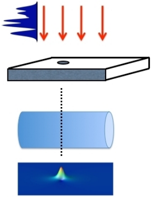


with $E(t) = \sum A_i \exp\left(-\left(\frac{t-t_i}{\Delta t_i}\right)^2\right) \cos(\omega_i t + \phi_i)$

Nanooptics: Test Case

Cooperation with T. Meier, M. Reichelt, Dep. Physik, Uni Paderborn

Generic configuration:



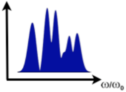
$\longleftarrow$ adaptable light puls $E(t)$

$\longleftarrow$ metal aperture with air hole

$\longleftarrow$ quantum wire

$\longleftarrow$ electron distribution?

Light puls:



with $E(t) = \sum A_i \exp\left(-\left(\frac{t-t_i}{\Delta t_i}\right)^2\right)\cos(\omega_i t + \phi_i)$

Parameter: A_i, ϕ_i, ω_i, t_i $\quad\Rightarrow$ up to 120!

Mathematical Formulation

State equation:

$$\begin{aligned}
\frac{\partial}{\partial t}p &= \frac{i}{\hbar}(\epsilon_0 - \epsilon_1)p + \frac{i}{\hbar}\mathbf{E}(t)\cdot\mathbf{d}\,(n_0 - n_1) \\
\frac{\partial}{\partial t}n_0 &= \frac{2}{\hbar}\mathrm{Im}\left[\mathbf{E}(t)\cdot\mathbf{d}\,p^*\right] \\
\frac{\partial}{\partial t}n_1 &= -\frac{2}{\hbar}\mathrm{Im}\left[\mathbf{E}(t)\cdot\mathbf{d}\,p^*\right] \\
1 &= n_1 + n_0
\end{aligned}$$

$\Rightarrow$ Three complex-valued coupled differential equations
p, n_0 and n_1 distributed in space.

Mathematical Formulation

State equation:

$$\begin{aligned}
\frac{\partial}{\partial t} p &= \frac{i}{\hbar}(\epsilon_0 - \epsilon_1)p + \frac{i}{\hbar}\mathbf{E}(t) \cdot \mathbf{d}\,(n_0 - n_1) \\
\frac{\partial}{\partial t} n_0 &= \frac{2}{\hbar}\mathrm{Im}\,[\mathbf{E}(t) \cdot \mathbf{d}\, p^*] \\
1 &= n_1 + n_0
\end{aligned}$$

Mathematical Formulation

State equation:

$$\frac{\partial}{\partial t} p = \frac{i}{\hbar}(\epsilon_0 - \epsilon_1)p + \frac{i}{\hbar}\mathbf{E}(t)\cdot\mathbf{d}\,(n_0 - n_1)$$

$$\frac{\partial}{\partial t} n_0 = \frac{2}{\hbar}\mathrm{Im}\,[\mathbf{E}(t)\cdot\mathbf{d}\,p^*]$$

$$1 = n_1 + n_0$$

Target:
Maximize energy at given time and given place with constant energy

= Maximize emitted radiation

$$I_{rad} = \left|\omega^2 P(\omega)\right|^2 = \left|\omega^2\, 2\,\mathrm{Re}(d\,p)\right|^2$$

Mathematical Setup

```
x[] ← (phase[], amplitude[], width[], point[])

for time=0 to Tfinal do
    if (time >= Tobs && time < Tobs+dt)
        eval_time_step1(x,int_tar)
    else
        eval_time_step2(x,int_tar)
    end if
end for

eval_target(int_tar,fitness)
```

Mathematical Setup

```
x[] ← (phase[], amplitude[], width[], point[])

for time=0 to Tfinal do
    if (time >= Tobs && time < Tobs+dt)
        eval_time_step1(x,int_tar)
    else
        eval_time_step2(x,int_tar)
    end if
end for

eval_target(int_tar,fitness)
```

currently:
independents $\in \{20, 60, 120\}$ $\Rightarrow$ Reverse mode!
time steps $\in \{16000, 32000, 160000\}$ $\Rightarrow$ 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!

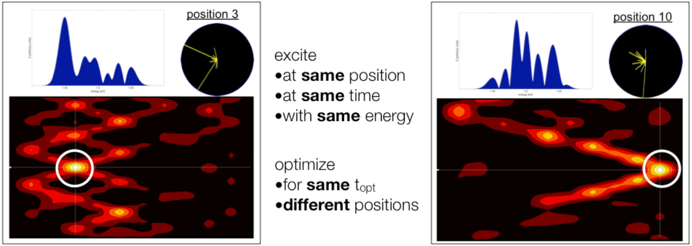
Nanooptics: Optimisation

So far: Genetic algorithms

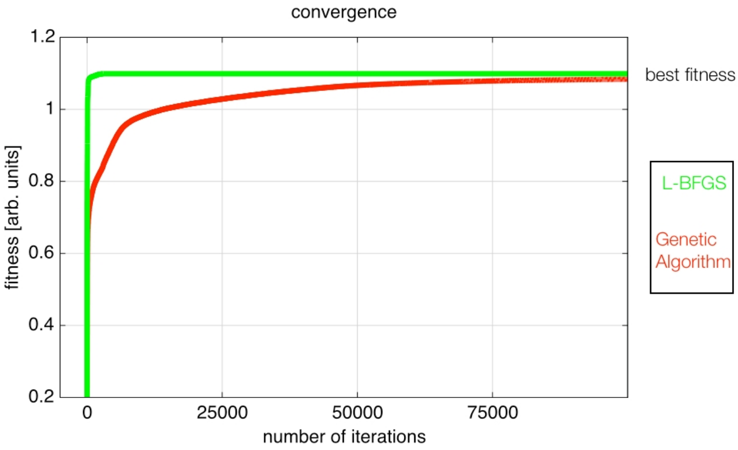
Now: L-BFGS and efficient gradient computation

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

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



Nanooptics: Comparison



(Walther, Reichelt, Meier 2011)

Conclusions

- Basics of Algorithmic Differentiation
  - Efficient evaluation of derivatives with working accuracy
  - Discrete Analogons of sensitivity and adjoint equation
  - Theory for basic modes complete, advanced AD?

Conclusions

- Basics of Algorithmic Differentiation
  - Efficient evaluation of derivatives with working accuracy
  - Discrete Analogons of sensitivity and adjoint equation
  - Theory for basic modes complete, advanced AD?
- Structure exploitation indispensable

Conclusions

- Basics of Algorithmic Differentiation
  - Efficient evaluation of derivatives with working accuracy
  - Discrete Analogons of sensitivity and adjoint equation
  - Theory for basic modes complete, advanced AD?
- Structure exploitation indispensable
- Consistent adjoint information? Efficient implementation?

  Suitable combination of continuous and discrete approach!