Introduction to Algorithmic Differentiation (AD)

Second-(and Higher-)Order Adjoint AD by Overloading ($F : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$)

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen

Contents

Outline

Newton's method approximates a solution to the unconstrained nonlinear optimization problem $\min_{x \in \mathbb{R}^n} f(x)$ through linearization of the first-order optimality condition

$$f'(x + \Delta x) = f'(x) + f''(x) \cdot \Delta x = 0$$

yielding systems of linear equations

$$f'' \cdot \Delta x = -f'$$

to be solved for Δx and followed by updating $x := x + \alpha \cdot \Delta x$ iteratively for a suitable start value x and damping parameter $\mathbb{R} \ni \alpha > 0$ determined by line search.

The objective function f is assumed to be twice continuously differentiable (implying $f''^T = f''$) at all points of interest.

Outline

Definition ($f : \boldsymbol{R}^n \to \boldsymbol{R}$)

A second derivative code generated in tangent of adjoint AD mode computes

$$\begin{pmatrix} y \\ y^{(2)} \\ \mathrm{x}_{(1)} \\ \mathrm{x}^{(2)}_{(1)} \end{pmatrix} = f^{(2)}_{(1)} \left(\mathrm{x}, \mathrm{x}^{(2)}, y_{(1)}, y^{(2)}_{(1)} \right)$$

as

$$\begin{pmatrix} y \\ y^{(2)} \\ \mathrm{x}_{(1)} \\ \mathrm{x}^{(2)}_{(1)} \end{pmatrix} = \begin{pmatrix} f(\mathrm{x}) \\ f' \cdot \mathrm{x}^{(2)} \\ y_{(1)} \cdot f' \\ \mathrm{x}^{(2)\,T} \cdot f'' \cdot y_{(1)} + y^{(2)}_{(1)} \cdot f' \end{pmatrix} ,$$

where $f' = f'(\mathrm{x}) \in \boldsymbol{R}^{1 \times n}$ and $f'' = f''(\mathrm{x}) \in \boldsymbol{R}^{n \times n}$.

Notes: nonincremental version; $\mathrm{x}_{(1)}, \mathrm{x}^{(2)}_{(1)} \in \boldsymbol{R}^{1 \times n}$; both $\mathrm{x}_{(1)}$ and $y^{(2)} \in \boldsymbol{R}$ required and non-vanishing $y^{(2)}_{(1)} \in \boldsymbol{R}$ in context of chain rule; $f''^{\,T} = f'' \in \boldsymbol{R}^{n \times n}$ as f twice continuously differentiable

Algorithmic differentiation of the (nonincremental) first-order adjoint model

$$\begin{pmatrix} y \\ x_{(1)} \end{pmatrix} = \begin{pmatrix} f(x) \\ y_{(1)} \cdot f' \end{pmatrix}$$

in tangent mode yields

$$\begin{pmatrix} y^{(2)} \\ x_{(1)}^{(2)\,T} \end{pmatrix} \equiv \frac{d\begin{pmatrix} y \\ x_{(1)}^T \end{pmatrix}}{d\begin{pmatrix} x \\ y_{(1)} \end{pmatrix}} \cdot \begin{pmatrix} x^{(2)} \\ y_{(1)}^{(2)} \end{pmatrix} = \begin{pmatrix} \frac{dy}{dx} \cdot x^{(2)} \left[+ \frac{dy}{dy_{(1)}} \cdot y_{(1)}^{(2)} = 0 \right] \\ \frac{dx_{(1)}^T}{dx} \cdot x^{(2)} + \frac{dx_{(1)}^T}{dy_{(1)}} \cdot y_{(1)}^{(2)} \end{pmatrix}$$

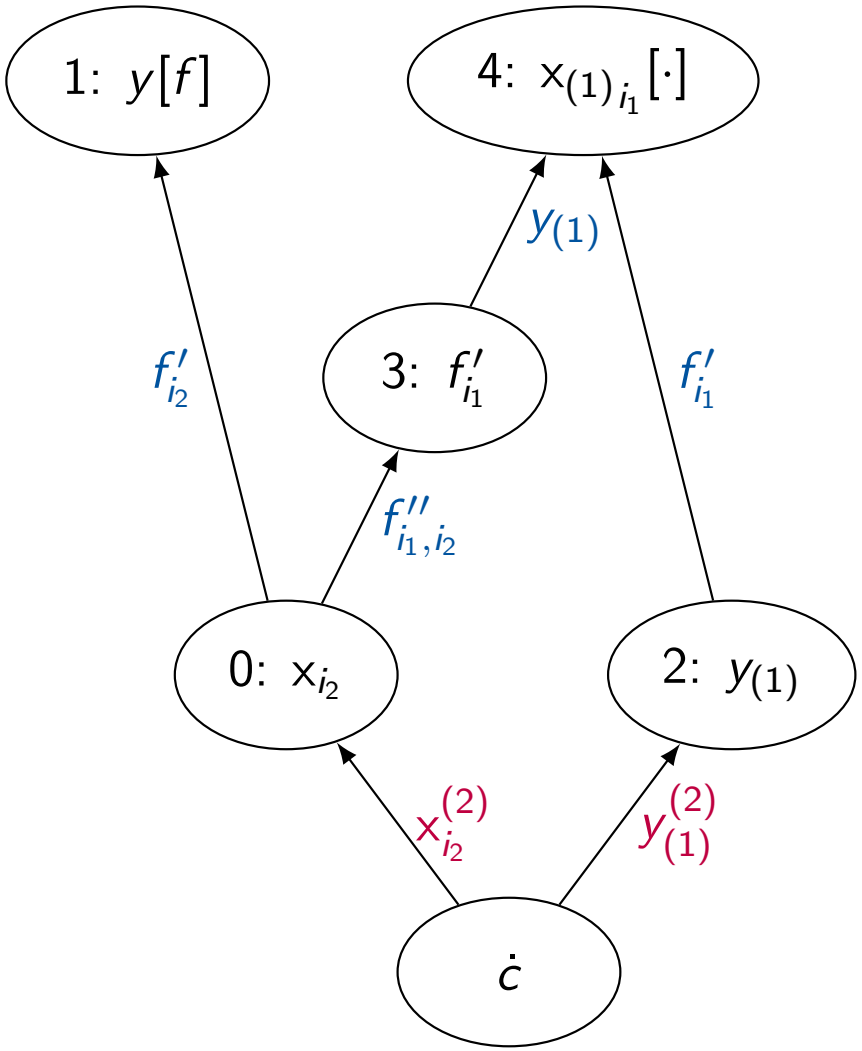
implying with[1] $x_{(1)}^T = f'^T \cdot y_{(1)}$ and $f''^T = f''$

$$\begin{pmatrix} y^{(2)} \\ x_{(1)}^{(2)} \end{pmatrix} = \begin{pmatrix} f' \cdot x^{(2)} \\ x^{(2)\,T} \cdot f'' \cdot y_{(1)} + y_{(1)}^{(2)} \cdot f' \end{pmatrix} .$$

[1] Note differentiation of column vectors rather than row vectors yielding Jacobians rather than transposed Jacobians for seamless application of the chain rule.

Tangents of Adjoints

Tangent of Adjoint DAG (using index notation)



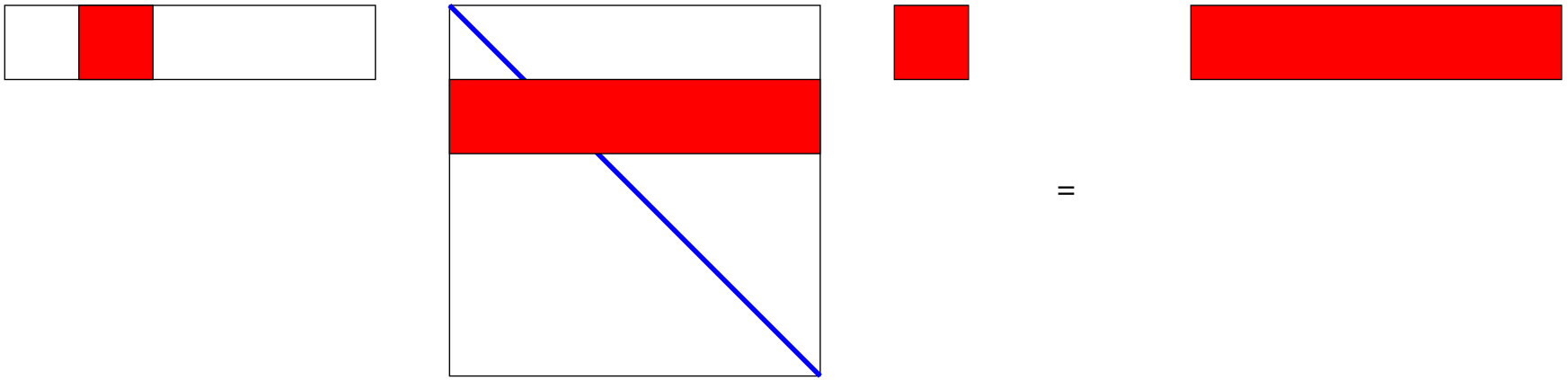
$$y^{(2)} \equiv \frac{dy}{d\dot{c}} = f'_{i_2} \cdot x^{(2)}_{i_2} = f' \cdot x^{(2)}$$

$$x^{(2)}_{(1)} \equiv \frac{dx_{(1)}}{d\dot{c}} = x^{(2)\,T} \cdot f'' \cdot y_{(1)} + y^{(2)}_{(1)} \cdot f'$$

$$x^{(2)}_{(1)i_1} = x^{(2)}_{i_2} \cdot f''_{i_1,i_2} \cdot y_{(1)} + y^{(2)}_{(1)} \cdot f'_{i_1}$$

Wake Up!

Draw the tangent of adjoint DAG for $y = \frac{x_0}{x_1}$.



$$x_{(1)}^{(2)} = x^{(2)^T} \cdot f'' \cdot y_{(1)} + y_{(1)}^{(2)} \cdot f'$$

... accumulation of the whole Hessian row-wise by seeding input directions $x^{(2)} \in \boldsymbol{R}^n$ with the Cartesian basis vectors in $\boldsymbol{R}^n$ for $y_{(1)} = 1$ and $y_{(1)}^{(2)} = 0$; harvesting from $x_{(1)}^{(2)}$.

For $j = n, \ldots, n+q-1$:

$$v_j = \varphi_j(v_k)_{k \prec j}$$
$$v_j^{(2)} = 0; \quad v_j^{(2)} \mathrel{+}= (\varphi_j')_{i_2} \cdot v_{i_2}^{(2)} \quad \forall\, i_2 \prec j\ .$$

For $j = n, \ldots, n+p-1$:

$$v_{j(1)} = 0; \quad v_{j(1)}^{(2)} = 0$$

For $j = n+q-1, \ldots, n$:

$$v_{i_1(1)} \mathrel{+}= v_{j(1)} \cdot (\varphi_j')_{i_1} \quad \forall i_1 \prec j$$
$$v_{i_1(1)}^{(2)} \mathrel{+}= v_{i_1}^{(2)} \cdot (\varphi_j'')_{i_1,i_2} \cdot v_{j(1)} + v_{j(1)}^{(2)} \cdot (\varphi_j')_{i_1} \quad \forall i_1, i_2 \prec j\ .$$

Tangent of Adjoint SAC

Example: $y = e^{\sin(x_0^2+x_1^2)}$

Seeding $v_{6(1)} = y_{(1)} = 1$ and a Cartesian basis vector, e.g. $v_0^{(2)} = x_0^{(2)} = 1$ and $v_1^{(2)} = x_1^{(2)} = 0$, while initializing all remaining inputs to zero and followed by propagating second-order adjoints alongside first-order adjoints (and primal function values) as

$$
\begin{aligned}
v_{5(1)} &+= v_6 \cdot v_{6(1)} & {v_5}_{(1)}^{(2)} &+= v_6^{(2)} \cdot v_{6(1)} + v_6 \cdot {v_6}_{(1)}^{(2)} \\
v_{4(1)} &+= \cos(v_4) \cdot v_{5(1)} & {v_4}_{(1)}^{(2)} &+= -\sin(v_4) \cdot v_{5(1)} \cdot v_4^{(2)} + \cos(v_4) \cdot {v_5}_{(1)}^{(2)} \\
v_{3(1)} &+= v_{4(1)} & {v_3}_{(1)}^{(2)} &+= {v_4}_{(1)}^{(2)} \\
v_{2(1)} &+= v_{4(1)} & {v_2}_{(1)}^{(2)} &+= {v_4}_{(1)}^{(2)} \\
v_{1(1)} &+= 2 \cdot v_1 \cdot v_{3(1)} & {v_1}_{(1)}^{(2)} &+= 2 \cdot (v_1^{(2)} \cdot v_{3(1)} + v_1 \cdot {v_3}_{(1)}^{(2)}) \\
v_{0(1)} &+= 2 \cdot v_0 \cdot v_{2(1)} & {v_0}_{(1)}^{(2)} &+= 2 \cdot (v_0^{(2)} \cdot v_{2(1)} + v_0 \cdot {v_2}_{(1)}^{(2)})
\end{aligned}
$$

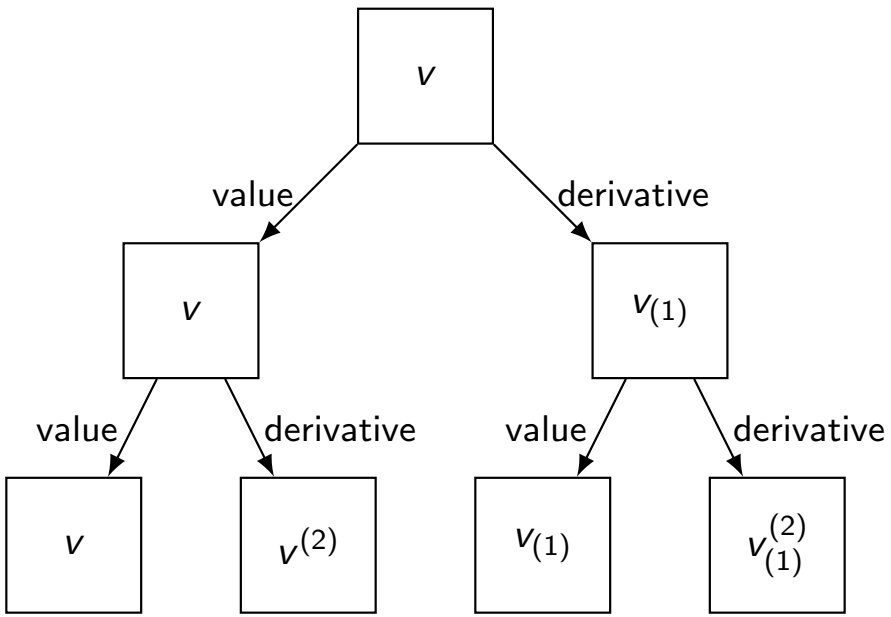
yields the corresponding row (here first) of the Hessian in $\begin{pmatrix} {v_0}_{(1)}^{(2)} & {v_1}_{(1)}^{(2)} \end{pmatrix}$;

omitting computation of v_i and $v_i^{(2)}$.

Tangents of Adjoints with dco/c++

Cheat Sheet

$$\begin{pmatrix} y \\ y^{(2)} \\ x_{(1)} \\ x^{(2)}_{(1)} \end{pmatrix} = \begin{pmatrix} f(x) \\ f' \cdot x^{(2)} \\ y_{(1)} \cdot f' \\ x^{(2)\,T} \cdot f'' \cdot y_{(1)} + y^{(2)}_{(1)} \cdot f' \end{pmatrix}$$




```
dco::value(dco::value(v))
[dco::passive_value(v)]
dco::derivative(dco::value(v))
dco::value(dco::derivative(v)
dco::derivative(dco::derivative(v)
```

```
1  template<typename T, typename PT>
2  void f(size_t ncp, size_t ncs, const PT &eps, T& x, const std::vector<T>& p, const
       std::vector<std::vector<PT>>& dW) {
3    newton(x,p[0],eps);
4    T s=0;
5    size_t m=dW.size();
6    for (size_t j=0;j<m;j+=ncp) paths(ncs,j,j+ncp,x,p,dW,s);
7    x=s/m;
8  }
```

We are looking for the Hessian of the final `x` with respect to `p`.

Hello AD World and dco/c++

$$\mathrm{p}^{(2)}_{(1)} = \mathrm{p}^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1
2
3   template<typename T, typename PT>
4   void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
      p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
      vector<std::vector<T>> &ddxdpp) {
5
6
7
8
9
10
11
12
13
14
15      ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6
7
8
9
10
11
12
13
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7
8
9
10
11
12
13
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape
9    using DCO_TPT=typename DCO_TT::position_t; // tape position
10
11
12
13
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape
9    using DCO_TPT=typename DCO_TT::position_t; // tape position
10   size_t n=p_v.size(); // size of gradient
11
12
13
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape
9    using DCO_TPT=typename DCO_TT::position_t; // tape position
10   size_t n=p_v.size(); // size of gradient
11   DCO_T x; std::vector<DCO_T> p(n); dco::passive_value(p)=p_v; // activate
12
13
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$

Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape
9    using DCO_TPT=typename DCO_TT::position_t; // tape position
10   size_t n=p_v.size(); // size of gradient
11   DCO_T x; std::vector<DCO_T> p(n); dco::passive_value(p)=p_v; // activate
12   DCO_M::global_tape=DCO_TT::create(); // "touch" tape
13
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$

Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape
9    using DCO_TPT=typename DCO_TT::position_t; // tape position
10   size_t n=p_v.size(); // size of gradient
11   DCO_T x; std::vector<DCO_T> p(n); dco::passive_value(p)=p_v; // activate
12   DCO_M::global_tape=DCO_TT::create(); // "touch" tape
13   DCO_M::global_tape->register_variable(p); // record active input
14
15   ...
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$

Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

```
1  #include "dco.hpp" // dco/c++
2
3  template<typename T, typename PT>
4  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, const std::vector<T> &
     p_v, const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp, std::
     vector<std::vector<T>> &ddxdpp) {
5    using DCO_BT=typename dco::gt1s<T>::type; // base type = tangent
6    using DCO_M=typename dco::ga1s<DCO_BT>; // base type of adjoint (mode)
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape
9    using DCO_TPT=typename DCO_TT::position_t; // tape position
10   size_t n=p_v.size(); // size of gradient
11   DCO_T x; std::vector<DCO_T> p(n); dco::passive_value(p)=p_v; // activate
12   DCO_M::global_tape=DCO_TT::create(); // "touch" tape
13   DCO_M::global_tape->register_variable(p); // record active input
14   DCO_TPT tpos=DCO_M::global_tape->get_position(); // store tape position
15   ...
```

Hello AD World and dco/c++

$$\mathrm{p}_{(1)}^{(2)} = \mathrm{p}^{(2)\,T} \cdot f'' \cdot x_{(1)} + x_{(1)}^{(2)} \cdot f'$$

Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

```
1     ...
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4
5
6
7
8
9
10
11
12
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5
6
7
8
9
10
11
12
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6
7
8
9
10
11
12
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7
8
9
10
11
12
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8
9
10
11
12
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8       for (size_t j=0;j<n;j++) {
9         ddxdpp[i][j]=dco::derivative(dco::derivative(p[j])); // harvest
10
11
12      }
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8       for (size_t j=0;j<n;j++) {
9         ddxdpp[i][j]=dco::derivative(dco::derivative(p[j])); // harvest
10        dco::derivative(dco::derivative(p[j]))=0; // reset
11        dco::value(dco::derivative(p[j]))=0;
12      }
13
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8       for (size_t j=0;j<n;j++) {
9         ddxdpp[i][j]=dco::derivative(dco::derivative(p[j])); // harvest
10        dco::derivative(dco::derivative(p[j]))=0; // reset
11        dco::value(dco::derivative(p[j]))=0;
12      }
13      dxdp[i]=dco::derivative(dco::value(x)); // harvest
14
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8       for (size_t j=0;j<n;j++) {
9         ddxdpp[i][j]=dco::derivative(dco::derivative(p[j])); // harvest
10        dco::derivative(dco::derivative(p[j]))=0; // reset
11        dco::value(dco::derivative(p[j]))=0;
12      }
13      dxdp[i]=dco::derivative(dco::value(x)); // harvest
14      dco::derivative(dco::value(p[i]))=0; // reset
15    }
16
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8       for (size_t j=0;j<n;j++) {
9         ddxdpp[i][j]=dco::derivative(dco::derivative(p[j])); // harvest
10        dco::derivative(dco::derivative(p[j]))=0; // reset
11        dco::value(dco::derivative(p[j]))=0;
12      }
13      dxdp[i]=dco::derivative(dco::value(x)); // harvest
14      dco::derivative(dco::value(p[i]))=0; // reset
15    }
16    x_v=dco::passive_value(x); // get primal
17
18  }
```

Hello AD World and dco/c++

$$p^{(2)}_{(1)} = p^{(2)^T} \cdot f'' \cdot x_{(1)} + x^{(2)}_{(1)} \cdot f'$$


```
1     ...
2     for (size_t i=0;i<n;i++) { // columns
3       dco::derivative(dco::value(p[i]))=1; // seed tangent
4       x=x_v; // [re]set
5       f(ncp,ncs,eps,x,p,dW); // compute
6       dco::value(dco::derivative(x))=1; // seed adjoint
7       DCO_M::global_tape->interpret_adjoint_and_reset_to(tpos); // interpret
8       for (size_t j=0;j<n;j++) {
9         ddxdpp[i][j]=dco::derivative(dco::derivative(p[j])); // harvest
10        dco::derivative(dco::derivative(p[j]))=0; // reset
11        dco::value(dco::derivative(p[j]))=0;
12      }
13      dxdp[i]=dco::derivative(dco::value(x)); // harvest
14      dco::derivative(dco::value(p[i]))=0; // reset
15    }
16    x_v=dco::passive_value(x); // get primal
17    DCO_TT::remove(DCO_M::global_tape); // deallocate tape
18  }
```

Given a (adjoint mode) gradient driver

```
1 template<typename T, typename PT>
2 void gradient(size_t ncp, size_t ncs, const PT &eps, T &x_v, std::vector<T> &p_v,
      const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp);
```

for

```
1 template<typename T, typename PT>
2 void f(size_t ncp, size_t ncs, const PT &eps, T& x, const std::vector<T>& p, const
      std::vector<std::vector<PT>>& dW);
```

the Hessian of the final `x` with respect to `p` can be computed as the Jacobian of the corresponding gradient.

Hello AD World and dco/c++

Jacobian of Gradient + Gradient + Function Value


```
1  template<typename T, typename PT> // Hessian driver
2  void hessian(size_t ncp, size_t ncs, const PT &eps, T &x_v, std::vector<T> &p_v,
     const std::vector<std::vector<PT>> &dW, std::vector<T> &dxdp_v, std::vector<
     std::vector<T>> &ddxdpp) {
3    using DCO_T=typename dco::gt1s<T>::type; // tangent type
4    size_t n=p_v.size(); // number of rows/columns in Hessian
5    DCO_T x; std::vector<DCO_T> p(n), dxdp(n); dco::value(p)=p_v; // activate
6    for (size_t i=0;i<n;i++) { // iterate over Cartesian basis
7      x=x_v; // [re]set inoutput
8      dco::derivative(p[i])=1; // seed
9      gradient(ncp,ncs,eps,x,p,dW,dxdp); // compute gradient
10     dxdp_v[i]=dco::value(dxdp[i]); // harvest gradient
11     for (size_t j=0;j<n;j++)
12       ddxdpp[i][j]=dco::derivative(dxdp[j]); // harvest Hessian
13     dco::derivative(p[i])=0; // unseed read-only input
14   }
15   x_v=dco::value(x); // get primal function value
16 }
```

Definition ($f : \boldsymbol{R}^n \to \boldsymbol{R}$)

A second derivative code generated in adjoint of tangent AD mode computes

$$\begin{pmatrix} y \\ y^{(1)} \\ x_{(2)} \\ x^{(1)}_{(2)} \end{pmatrix} = f^{(1)}_{(2)} \left(x, x^{(1)}, y_{(2)}, y^{(1)}_{(2)} \right)$$

as

$$\begin{pmatrix} y \\ y^{(1)} \\ x_{(2)} \\ x^{(1)}_{(2)} \end{pmatrix} = \begin{pmatrix} f(x) \\ f' \cdot x^{(1)} \\ y_{(2)} \cdot f' + y^{(1)}_{(2)} \cdot {x^{(1)}}^T \cdot f'' \\ y^{(1)}_{(2)} \cdot f' \end{pmatrix},$$

where $f' = f'(x) \in \boldsymbol{R}^{1 \times n}$ and $f'' = f''(x) \in \boldsymbol{R}^{n \times n}$.

Notes: nonincremental version; $x_{(2)}, x^{(1)}_{(2)} \in \boldsymbol{R}^{1 \times n}$; both $x^{(1)}_{(2)}$ and $y^{(1)} \in \boldsymbol{R}$ required and non-vanishing $y_{(2)} \in \boldsymbol{R}$ in context of chain rule; ${f''}^T = f'' \in \boldsymbol{R}^{n \times n}$ as f twice continuously differentiable

Algorithmic differentiation of the (nonincremental) first-order tangent model

$$\begin{pmatrix} y \\ y^{(1)} \end{pmatrix} = \begin{pmatrix} f(x) \\ f' \cdot x^{(1)} \end{pmatrix}$$

in adjoint mode yields

$$\begin{pmatrix} x_{(2)} & x^{(1)}_{(2)} \end{pmatrix} \equiv \begin{pmatrix} y_{(2)} & y^{(1)}_{(2)} \end{pmatrix} \cdot \frac{d \begin{pmatrix} y \\ y^{(1)} \end{pmatrix}}{d \begin{pmatrix} x \\ x^{(1)} \end{pmatrix}}$$

$$= \left(y_{(2)} \cdot \frac{dy}{dx} + y^{(1)}_{(2)} \cdot \frac{dy^{(1)}}{dx} \quad \left[y_{(2)} \cdot \frac{dy}{dx^{(1)}} = 0 \right] + y^{(1)}_{(2)} \cdot \frac{dy^{(1)}}{dx^{(1)}} \right)$$

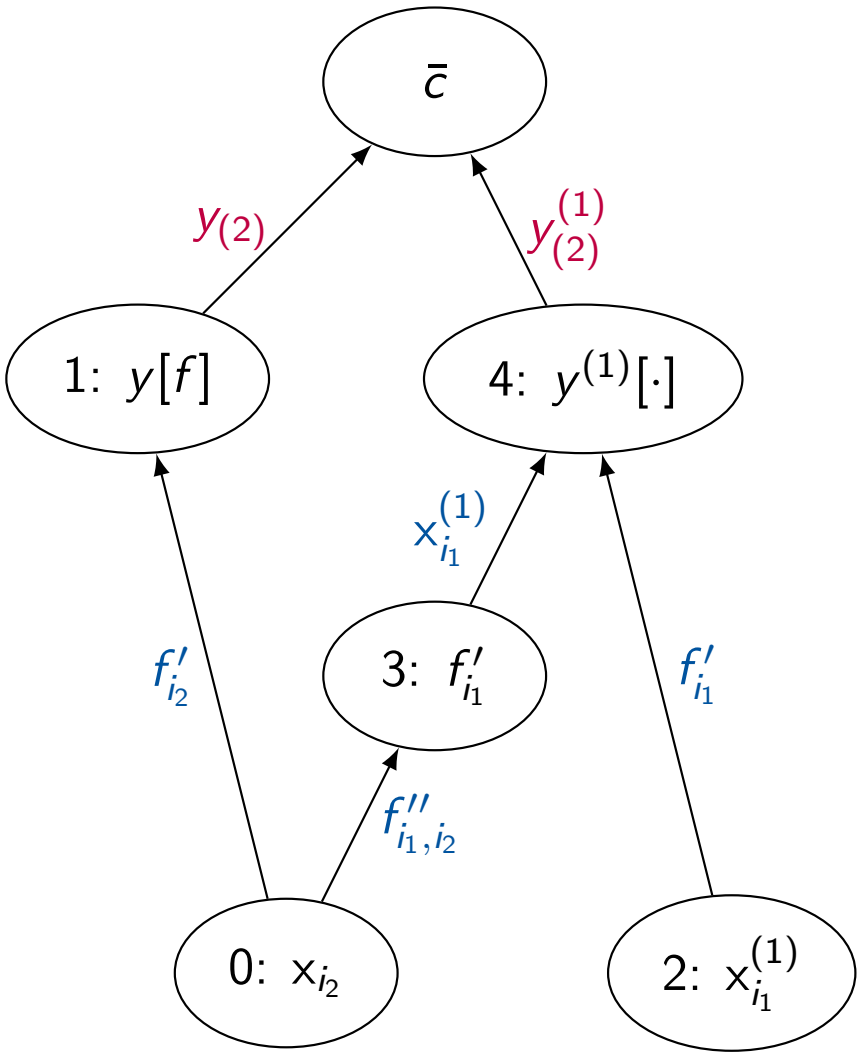
implying with[2] $y^{(1)} = {x^{(1)}}^T \cdot {f'}^T$ and ${f''}^T = f''$

$$\begin{pmatrix} x_{(2)} & x^{(1)}_{(2)} \end{pmatrix} = \begin{pmatrix} y_{(2)} \cdot f' + y^{(1)}_{(2)} \cdot {x^{(1)}}^T \cdot f'' & y^{(1)}_{(2)} \cdot f' \end{pmatrix}$$

[2] Note differentiation of column vectors rather than row vectors yielding Jacobians rather than transposed Jacobians for seamless application of the chain rule.

Adjoints of Tangents

Adjoint of Tangent DAG (using index notation)



$$x^{(1)}_{(2)} \equiv \frac{d\bar{c}}{dx^{(1)}} = y^{(1)}_{(2)} \cdot f'$$

$$x^{(1)}_{(2)\,i_1} = y^{(1)}_{(2)} \cdot f'_{i_1}$$

$$x_{(2)} \equiv \frac{d\bar{c}}{dx} = y_{(2)} \cdot f' + y^{(1)}_{(2)} \cdot x^{(1)\,T} \cdot f''$$

$$x_{(2)\,i_2} = y_{(2)} \cdot f'_{i_2} + y^{(1)}_{(2)} \cdot x^{(1)}_{i_1} \cdot f''_{i_1,i_2}$$

Wake Up!

Draw the tangent of adjoint DAG for $y = x_0^2 + x_1^2$.

Definition ($f : \boldsymbol{R}^n \to \boldsymbol{R}$)

A second derivative code generated in adjoint of adjoint AD mode computes

$$\begin{pmatrix} y \\ \mathrm{x}_{(1)} \\ \mathrm{x}_{(2)} \\ y_{(1,2)} \end{pmatrix} = f_{(1,2)} \left(\mathrm{x}, \mathrm{x}^{(1,2)}, y_{(1)}, y_{(1,2)} \right)$$

as

$$\begin{pmatrix} y \\ \mathrm{x}_{(1)} \\ \mathrm{x}_{(2)} \\ y_{(1,2)} \end{pmatrix} = \begin{pmatrix} f(\mathrm{x}) \\ y_{(1)} \cdot f' \\ y_{(2)} \cdot f' + \mathrm{x}_{(1,2)}^T \cdot y_{(1)} \cdot f'' \\ f' \cdot \mathrm{x}_{(1,2)} \ , \end{pmatrix}$$

where $f' = f'(\mathrm{x}) \in \boldsymbol{R}^{1 \times n}$ and $f'' = f''(\mathrm{x}) \in \boldsymbol{R}^{n \times n}$.

Notes: nonincremental version; $\mathrm{x}_{(1)}, \mathrm{x}_{(2)} \in \boldsymbol{R}^{1 \times n}$, $\mathrm{x}_{(1,2)} \in \boldsymbol{R}^n$; both $\mathrm{x}_{(1)}$ and $y_{(1,2)} \in \boldsymbol{R}$ required and non-vanishing $y_{(2)} \in \boldsymbol{R}$ in context of chain rule; $f''^T = f'' \in \boldsymbol{R}^{n \times n}$ as f twice continuously differentiable

Algorithmic differentiation of the (nonincremental) first-order adjoint model

$$\begin{pmatrix} y \\ x_{(1)} \end{pmatrix} = \begin{pmatrix} f(x) \\ y_{(1)} \cdot f' \end{pmatrix}$$

in adjoint mode yields

$$\begin{pmatrix} x_{(2)} & y_{(1,2)} \end{pmatrix} \equiv \begin{pmatrix} y_{(2)} & x_{(1,2)}^T \end{pmatrix} \cdot \frac{d \begin{pmatrix} y \\ x_{(1)}^T \end{pmatrix}}{d \begin{pmatrix} x \\ y_{(1)} \end{pmatrix}}$$

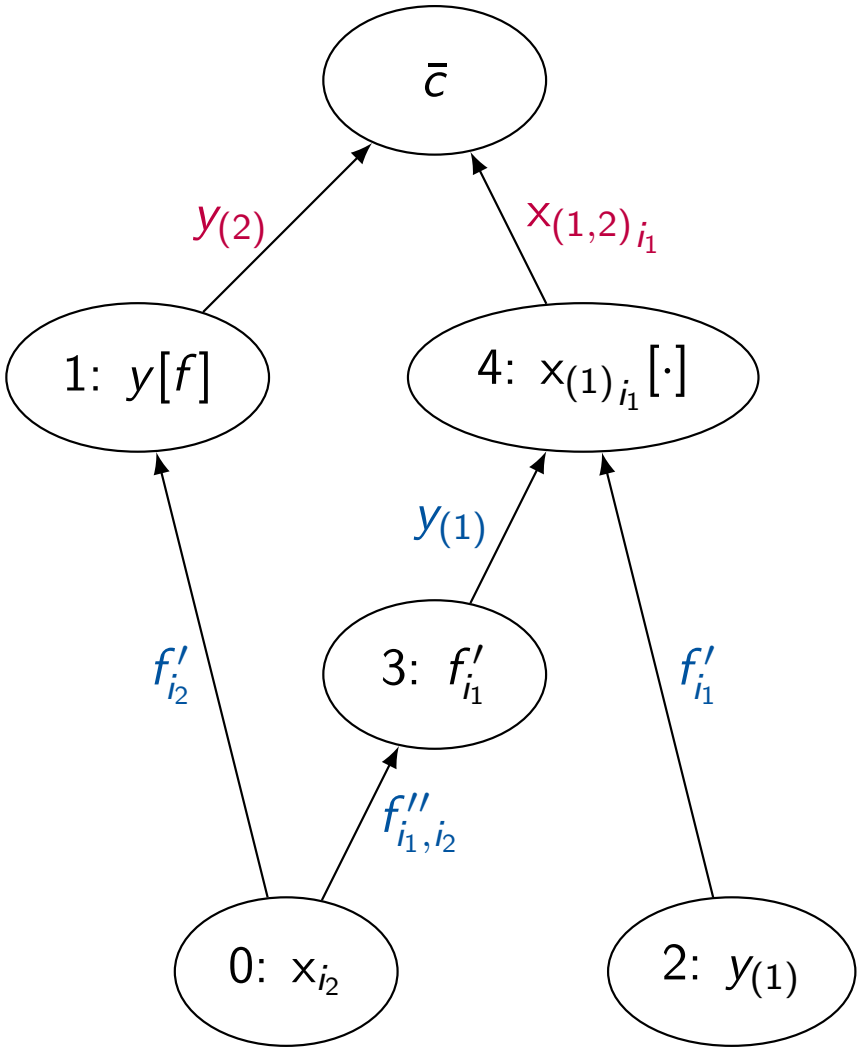
$$= \left(y_{(2)} \cdot \frac{dy}{dx} + x_{(1,2)}^T \cdot \frac{dx_{(1)}^T}{dx} \quad \left[y_{(2)} \cdot \frac{dy}{dy_{(1)}} = 0 \right] + x_{(1,2)}^T \cdot \frac{dx_{(1)}^T}{dy_{(1)}} \right)$$

implying with[3] ${x_{(1)}}^T = f'^T \cdot y_{(1)} = y_{(1)} \cdot f'^T$, $x_{(1,2)}^T \cdot f'^T = f' \cdot x_{(1,2)}$ and $f''^T = f''$

$$\begin{pmatrix} x_{(2)} & y_{(1,2)} \end{pmatrix} = \begin{pmatrix} y_{(2)} \cdot f' + x_{(1,2)}^T \cdot y_{(1)} \cdot f'' & f' \cdot x_{(1,2)} \end{pmatrix}$$

[3] Note $x_{(1,2)} \in \boldsymbol{R}^n$ and differentiation of column vectors rather than row vectors yielding Jacobians rather than transposed Jacobians for seamless application of chain rule.

Adjoint of Adjoint DAG



$$x_{(2)} \equiv \frac{d\bar{c}}{dx} = y_{(2)} \cdot f' + x_{(1,2)} \cdot y_{(1)} \cdot f''$$

$$x_{(2)\,i_2} = y_{(2)} \cdot f'_{i_2} + x_{(1,2)\,i_1} \cdot y_{(1)} \cdot f''_{i_1,i_2}$$

$$y_{(1,2)} \equiv \frac{d\bar{c}}{dy_{(1)}} = x^T_{(1,2)} \cdot f'^T = x_{(1,2)\,i_1} \cdot f'_{i_1}$$

Wake Up!

Draw the adjoint of adjoint DAG for $y = \sin(x_0) * x_1$.

Outline

$$F : \boldsymbol{R}^n \to \boldsymbol{R}^m : \quad y = F(\mathrm{x})$$

is assumed to be twice continuously differentiable with

Jacobian

$$F' = F'(\mathrm{x}) = \left(F'_{j,i}\right)_{i=0,\ldots,n-1}^{j=0,\ldots,m-1}$$

and Hessian

$$F'' = F''(\mathrm{x}) = \left(F''_{j,i_1,i_2}\right)_{i_1,i_2=0,\ldots,n-1}^{j=0,\ldots,m-1} .$$

Definition ($F : \boldsymbol{R}^n \to \boldsymbol{R}^m$)

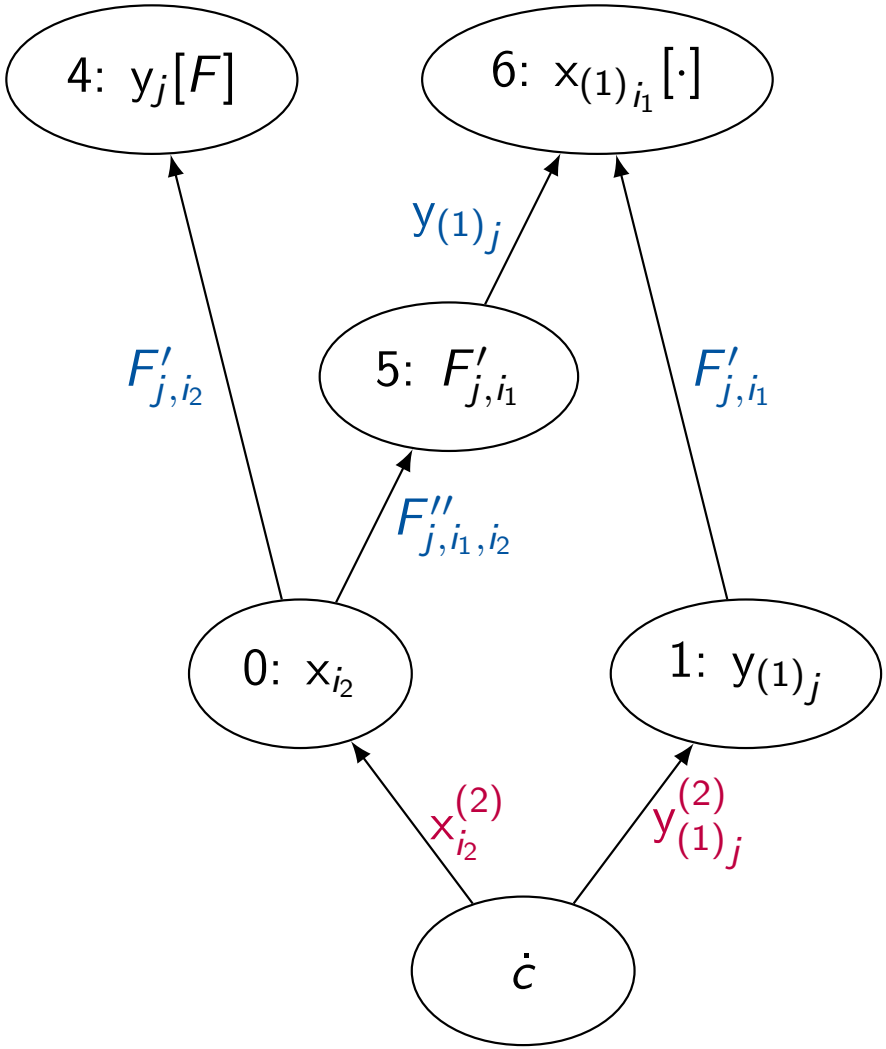
A second derivative code generated in tangent of adjoint AD mode computes

$$\begin{pmatrix} \mathrm{y} \\ \mathrm{y}_{(1)}^{(2)} \\ \mathrm{x}_{(1)} \\ \mathrm{x}_{(1)}^{(2)} \end{pmatrix} = F_{(1)}^{(2)} \left(\mathrm{x}, \mathrm{x}^{(2)}, \mathrm{y}_{(1)}, \mathrm{y}_{(1)}^{(2)} \right)$$

as

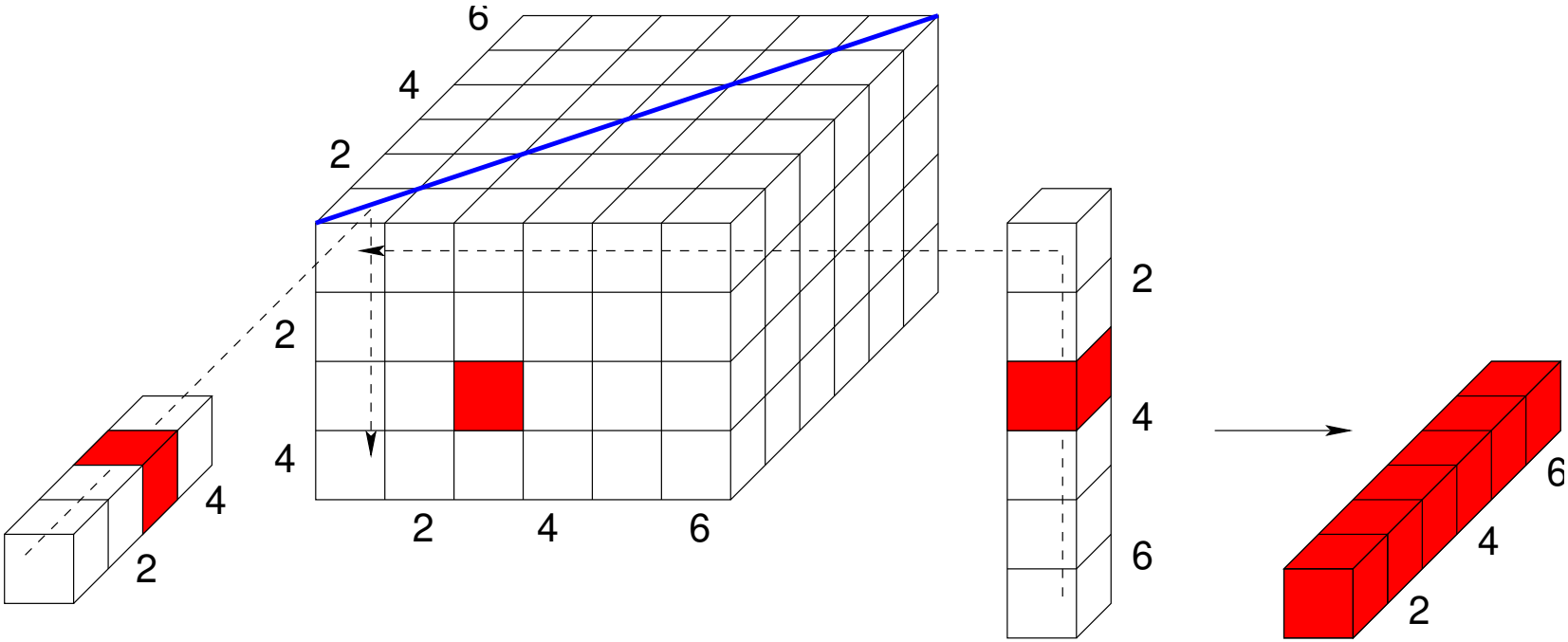
$$\begin{pmatrix} \mathrm{y} \\ \mathrm{y}_j^{(2)} \\ \mathrm{x}_{(1)_{i_1}} \\ \mathrm{x}_{(1)_{i_1}}^{(2)} \end{pmatrix} = \begin{pmatrix} F(\mathrm{x}) \\ F'_{j,i_2} \cdot \mathrm{x}_{i_2}^{(2)} \\ \mathrm{y}_{(1)_j} \cdot F'_{j,i_1} \\ \mathrm{x}_{i_2}^{(2)} \cdot F''_{j,i_1,i_2} \cdot \mathrm{y}_{(1)_j} + \mathrm{y}_{(1)_j}^{(2)} \cdot F'_{j,i_1} \end{pmatrix} .$$

Note: nonincremental version assumes initially vanishing $\mathrm{y}^{(2)}$, $\mathrm{x}_{(1)}$ and $\mathrm{x}_{(1)}^{(2)}$.



$$y^{(2)} \equiv \frac{dy}{d\dot{c}}$$
$$y^{(2)}_j = F'_{j,i_2} \cdot x^{(2)}_{i_2}$$

$$x^{(2)}_{(1)} \equiv \frac{dx_{(1)}}{d\dot{c}}$$
$$x^{(2)}_{(1)_{i_1}} = x^{(2)}_{i_2} \cdot F''_{j,i_1,i_2} \cdot y_{(1)_j} + y^{(2)}_{(1)_j} \cdot F'_{j,i_1}$$


6
4
2
2
4
2
4
6
4
2
2
4
6
2
4
6

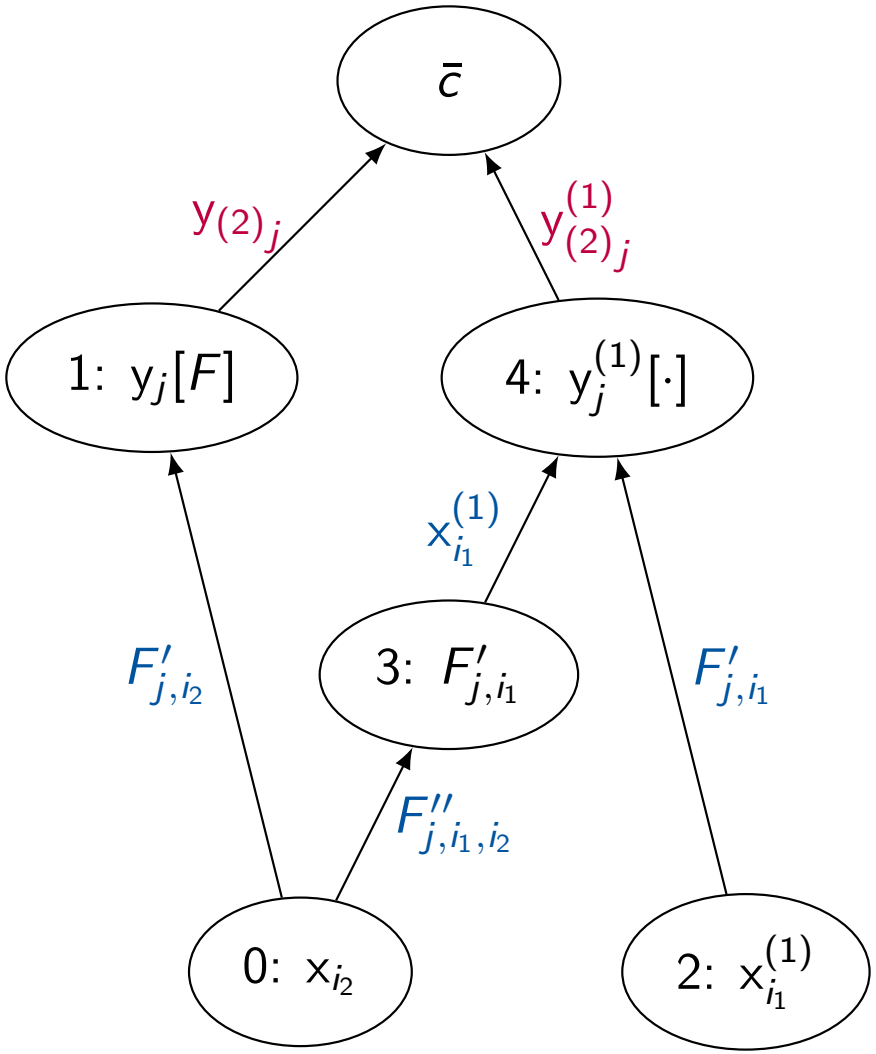
A second derivative code generated in adjoint of tangent AD mode computes

$$\begin{pmatrix} y \\ y^{(1)} \\ x_{(2)} \\ x^{(1)}_{(2)} \end{pmatrix} = F^{(1)}_{(2)} \left(x, x^{(1)}, y_{(2)}, y^{(1)}_{(2)} \right)$$

as

$$\begin{pmatrix} y \\ y^{(1)}_j \\ x_{(2)_{i_2}} \\ x^{(1)}_{(2)_{i_1}} \end{pmatrix} = \begin{pmatrix} F(x) \\ F'_{j,i_1} \cdot x^{(1)}_{i_1} \\ y_{(2)_j} \cdot F'_{j,i_2} + y^{(1)}_{(2)_j} \cdot x^{(1)}_{i_1} \cdot F''_{j,i_1,i_2} \\ y^{(1)}_{(2)_j} \cdot F'_{j,i_1} \end{pmatrix} .$$

Note: nonincremental version assumes initially vanishing $y^{(1)}$, $x_{(2)}$ and $x^{(1)}_{(2)}$.



$$x^{(1)}_{(2)} \equiv \frac{d\bar{c}}{dx^{(1)}}$$

$$x^{(1)}_{(2)_{i_1}} = y^{(1)}_{(2)_j} \cdot F'_{j,i_1}$$

$$x_{(2)} \equiv \frac{d\bar{c}}{dx}$$

$$x_{(2)_{i_2}} = y_{(2)_j} \cdot F'_{j,i_2} + y^{(1)}_{(2)_j} \cdot x^{(1)}_{i_1} \cdot F''_{j,i_1,i_2}$$

A second derivative code generated in adjoint of adjoint AD mode computes

$$\begin{pmatrix} y \\ x_{(1)} \\ x_{(2)} \\ y_{(1,2)} \end{pmatrix} = F_{(1,2)} \left(x, x^{(1,2)}, y_{(1)}, y_{(1,2)} \right)$$

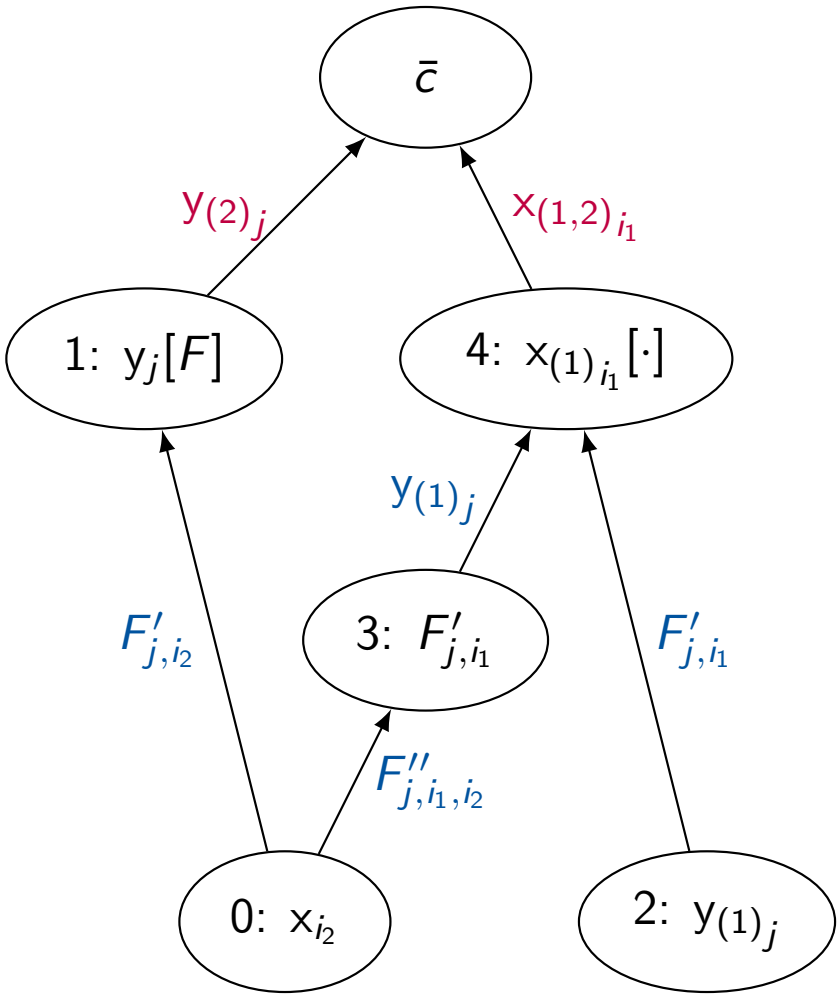
as

$$\begin{pmatrix} y \\ x_{(1)_{i_1}} \\ x_{(2)_{i_2}} \\ y_{(1,2)_j} \end{pmatrix} = \begin{pmatrix} F(x) \\ y_{(1)_j} \cdot F'_{j,i_1} \\ y_{(2)_j} \cdot F'_{j,i_2} + x_{(1,2)_{i_1}} \cdot y_{(1)_j} \cdot F''_{j,i_1,i_2} \\ x_{(1,2)_{i_1}} \cdot F'_{j,i_1} \end{pmatrix} .$$

Note: nonincremental version assumes initially vanishing $x_{(1)}$, $x_{(2)}$ and $y_{(1,2)}$.

Adjoints of Adjoints

Adjoint of Adjoint DAG



$$x_{(2)} \equiv \frac{d\bar{c}}{dx}$$

$$x_{(2)_{i_2}} = y_{(2)_j} \cdot F'_{j,i_2} + x_{(1,2)_{i_1}} \cdot y_{(1)_j} \cdot F''_{j,i_1,i_2}$$

$$y_{(1,2)} \equiv \frac{d\bar{c}}{dy_{(1)}}$$

$$y_{(1,2)_j} = x_{(1,2)_{i_1}} \cdot F'_{j,i_1}$$

Outline

$$
\begin{aligned}
y &= F(x) \\
y^{(3)}_j &= F'_{j,i_3} \cdot x^{(3)}_{i_3} \\
y^{(1)}_j &= F'_{j,i_1} \cdot x^{(1)}_{i_1} \\
y^{(1,3)}_j &= F''_{j,i_1,i_3} \cdot x^{(1)}_{i_1} \cdot x^{(3)}_{i_3} + F'_{j,i_1} \cdot x^{(1,3)}_{i_1} \\
x_{(2)\,i_2} &= y_{(2)\,j} \cdot F'_{j,i_2} + y^{(1)}_{(2)\,j} \cdot x^{(1)}_{i_1} \cdot F''_{j,i_1,i_2} \\
x^{(3)}_{(2)\,i_2} &= y^{(3)}_{(2)\,j} \cdot F'_{j,i_2} + y_{(2)\,j} \cdot F''_{j,i_2,i_3} \cdot x^{(3)}_{i_3} + y^{(1,3)}_{(2)\,j} \cdot x^{(1)}_{i_1} \cdot F''_{j,i_1,i_2} \\
&\quad + y^{(1)}_{(2)\,j} \cdot x^{(1,3)}_{i_1} \cdot F''_{j,i_1,i_2} + y^{(1)}_{(2)\,j} \cdot x^{(1)}_{i_1} \cdot F'''_{j,i_1,i_2,i_3} \cdot x^{(3)}_{i_3} \\
x^{(1)}_{(2)\,i_1} &= y^{(1)}_{(2)\,j} \cdot F'_{j,i_1} \\
x^{(1,3)}_{(2)\,i_1} &= y^{(1,3)}_{(2)\,j} \cdot F'_{j,i_1} + y^{(1)}_{(2)\,j} \cdot F''_{j,i_1,i_3} \cdot x^{(3)}_{i_3}
\end{aligned}
$$

Note commutativity of multiplication in index notation.

Wake Up!

Draw the adjoint of tangent of adjoint DAG.

$$x_{(2)}^{(3,4)} = ?$$

Outline

Summary