The Chain Rule of Differentiation is Associative.

So What?

Uwe Naumann

RWTH Aachen University, Aachen, Germany and NAG Ltd., Oxford, UK

The Chain Rule of Differentiation is Associative

Let

$$y = F(x) = f(g(h(x))) : \mathbb{R}^n \to \mathbb{R}^m \ .$$

If f, g, h are differentiable with Jacobians $f' = f'(g), g' = g'(h), h' = h'(x)$, then so is F and

$$F' = F'(x) = f' \cdot g' \cdot h' \ .$$

The Chain Rule of Differentiation is Associative

Let

$$y = F(x) = f(g(h(x))) : \mathbb{R}^n \to \mathbb{R}^m \ .$$

If f, g, h are differentiable with Jacobians $f' = f'(g), g' = g'(h), h' = h'(x)$, then so is F and

$$F' = F'(x) = f' \cdot g' \cdot h' \ .$$

Matrix multiplication is associative, that is, in infinite precision arithmetic

$$F' = f' \cdot (g' \cdot h') = (f' \cdot g') \cdot h' \ .$$

So What?

The order matters (for computational cost)!

Different orders induce different costs, e.g., for $a_i \in \mathbb{R}^n$ (Note $F : \mathbb{R} \to \mathbb{R}^n$)

$$a_5 \cdot a_4^T \cdot a_3 \cdot a_2^T \cdot a_1 = a_5 \cdot (a_4^T \cdot (a_3 \cdot (a_2^T \cdot a_1))) \qquad \Rightarrow \mathcal{O}(n)$$
$$| \, (\, - \, (\, | \, (\, -| \,) \,) \,)$$

So What?

The order matters (for computational cost)!

Different orders induce different costs, e.g., for $a_i \in \mathbb{R}^n$ (Note $F : \mathbb{R} \to \mathbb{R}^n$)

$$a_5 \cdot a_4^T \cdot a_3 \cdot a_2^T \cdot a_1 = a_5 \cdot (a_4^T \cdot (a_3 \cdot (a_2^T \cdot a_1))) \qquad \Rightarrow \mathcal{O}(n)$$

$$| \, (\, - \, (\, | \, (\, -| \,) \,) \,)$$

$$= (((a_5 \cdot a_4^T) \cdot a_3) \cdot a_2^T) \cdot a_1 \qquad \Rightarrow \mathcal{O}(n^2)$$

$$(\, (\, (\, |- \,) \, | \,) - \,) \, |$$

So What?

The order matters (for computational cost)!

Different orders induce different costs, e.g., for $a_i \in \mathbb{R}^n$ (Note $F : \mathbb{R} \to \mathbb{R}^n$)

$$a_5 \cdot a_4^T \cdot a_3 \cdot a_2^T \cdot a_1 = a_5 \cdot (a_4^T \cdot (a_3 \cdot (a_2^T \cdot a_1))) \qquad \Rightarrow \mathcal{O}(n)$$
$$| \, (\, - \, (\, | \, (\, - | \,) \,) \,)$$

$$= (((a_5 \cdot a_4^T) \cdot a_3) \cdot a_2^T) \cdot a_1 \qquad \Rightarrow \mathcal{O}(n^2)$$
$$(\, (\, (\, | - \,) \, | \,) - \,) \, |$$

$$= ((a_5 \cdot a_4^T) \cdot (a_3 \cdot a_2^T)) \cdot a_1 \qquad \Rightarrow \mathcal{O}(n^3) \, .$$
$$(\, (\, | - \,) \, (| - \,) \,) \, |$$

So What?

The order matters (for computational cost)!

Different orders induce different costs, e.g., for $a_i \in \mathbb{R}^n$ (Note $F : \mathbb{R} \to \mathbb{R}^n$)

$$a_5 \cdot a_4^T \cdot a_3 \cdot a_2^T \cdot a_1 = a_5 \cdot (a_4^T \cdot (a_3 \cdot (a_2^T \cdot a_1))) \qquad \Rightarrow \mathcal{O}(n)$$
$$| \, (\, - \, (\, | \, (\, - | \,) \,) \,)$$

$$= (((a_5 \cdot a_4^T) \cdot a_3) \cdot a_2^T) \cdot a_1 \qquad \Rightarrow \mathcal{O}(n^2)$$
$$(\, (\, (\, | - \,) \, | \,) - \,) \, |$$

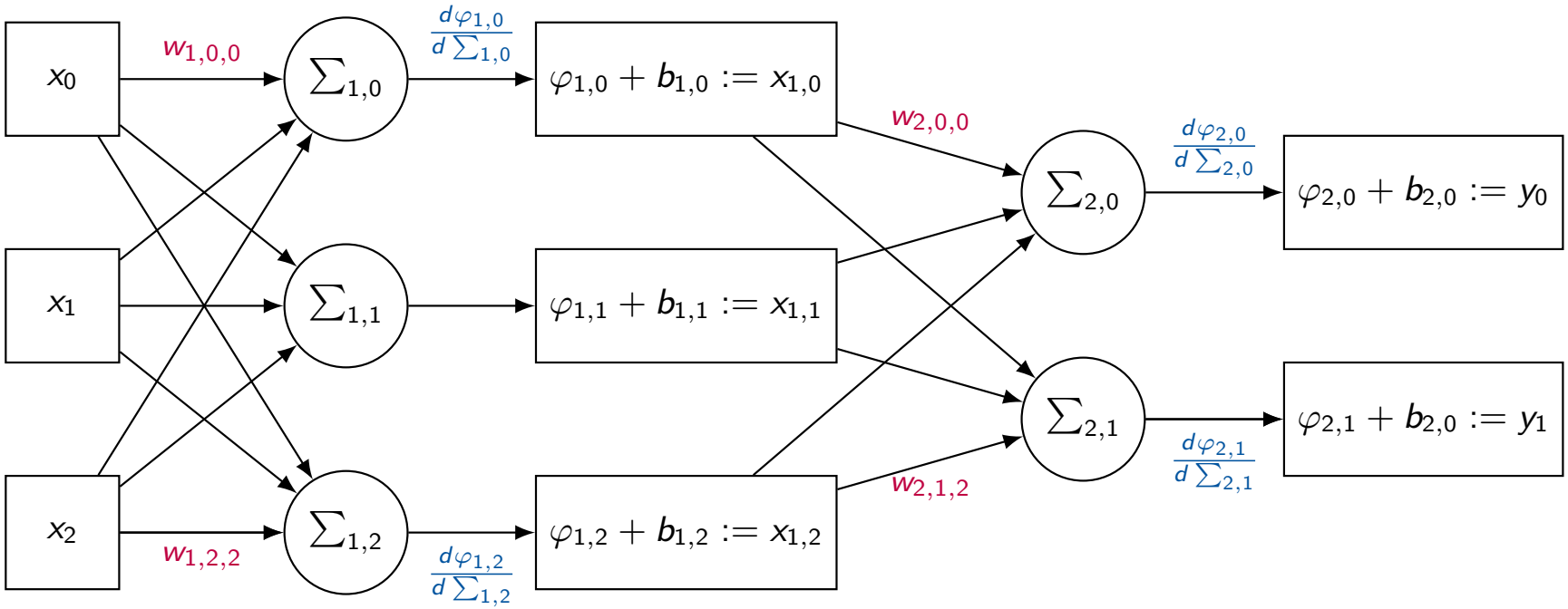
$$= ((a_5 \cdot a_4^T) \cdot (a_3 \cdot a_2^T)) \cdot a_1 \qquad \Rightarrow \mathcal{O}(n^3) \, .$$
$$(\, (\, | - \,) \, (| - \,) \,) \, |$$

... and what about numerical stability?

A. Griewank, K. Kulshreshtha and A. Walther: On the numerical stability of algorithmic differentiation. Computing 94, pages 125–149, 2012.

A Popular Special Case

Backpropagation in Artificial Neural Networks



... a special case of adjoint algorithmic differentiation, which aims for applicability of the same fundamental idea to general differentiable computer programs.

Contents

Outline

Introduction

- Member (since 2008) and Principal Scientist (2022), NAG Ltd., Oxford, UK
- Visiting Lecturer, University of Oxford, UK (2017)
- Professor of Computer Science, RWTH Aachen University, Aachen, Germany (since 2004)
- Assistant Computer Scientist, Argonne National Laboratory, Argonne, IL, US (2002–2004)
- Visiting Scientist, MIT, Cambridge, MA, US (2001)
- Senior Lecturer for Computer Science, University of Hertfordshire (UHerts), Hatfield, UK (2000–2001) $\Rightarrow$ Visiting Researcher
- Postdoctoral Researcher, INRIA, Sophia-Antipolis, France (1999–2000)
- MSc/PhD in Applied Mathematics, Technical University Dresden, Germany, University of York, UK, Chuo University, Tokyo. Japan, UHerts, UK (1990–1999, supervisor: Andreas Griewank, co-supervisor: Bruce Christianson)
- Military service in Bad Düben, East Germany (1988–1990)
- went to school in Chemnitz, East Germany and Dubna, Russia (1976–1988)

Outline

Let $y = F(x) : \mathbb{R}^n \to \mathbb{R}^m$ be implemented as a differentiable program.

Let $y = F(x) : \mathbb{R}^n \to \mathbb{R}^m$ be implemented as a differentiable program.

Tangent AD yields

$$\mathbb{R}^m \ni \dot{y} = \dot{F}(x, \dot{x}) = F'(x) \cdot \dot{x}$$

and hence F' at $\leq \mathcal{O}(n) \cdot \mathrm{ERT}(\dot{F})$.

Let $y = F(x) : \mathbb{R}^n \to \mathbb{R}^m$ be implemented as a differentiable program.

Tangent AD yields

$$\mathbb{R}^m \ni \dot{y} = \dot{F}(x, \dot{x}) = F'(x) \cdot \dot{x}$$

and hence F' at $\leq \mathcal{O}(n) \cdot \textsc{Ert}(\dot{F})$.

Associative chain rule $\Rightarrow$ Adjoint AD yields

$$\mathbb{R}^{1 \times n} \ni \bar{x} = \bar{F}(x, \bar{y}) = \bar{y} \cdot F'(x)$$

and hence F' at $\leq \mathcal{O}(m) \cdot \textsc{Ert}(\bar{F})$. ($\Rightarrow$ reversal of the data flow)

A. Griewank and A. Walther: Evaluating Derivatives. 2nd edition. SIAM 2008.

U.N.: The Art of Differentiating Computer Programs. SIAM 2012.

U.N.: DAG Reversal is NP-Complete. Journal of Discrete Algorithms, 7(4), 2009.

U.N.: Call Tree Reversal is NP-Complete. AD 2008.

NAG's dco/c++

Derivative Code by Overloading in C++

- efficient tangents and adjoints of arbitrary order for differentiable C++ programs through dco::g[t,a]1s<BASE_TYPE>::type
- highly flexible API supports AD beyond the "black box"
- extensive training and support
- users include various tier-1 investment banks (e.g. Scotia Bank received Risk.net's 2021 Technology Innovation of the Year award) and Formula 1 teams

Refer to appendix for sample code.

K. Leppkes, J. Lotz and U.N.: Derivative Code by Overloading in C++. NAG TR2/20, 2020.

U.N. and J. du Toit: Adjoint Algorithmic Differentiation Tool Support for Typical Numerical Patterns in Computational Finance. Journal of Computational Finance 21(4), 2018.

Outline

Given:

- a modular differentiable program F consisting of elemental functions (modules) F_i, $i = 1, \ldots, p$
- tangents $\dot{F}_i$ ($\rightarrow$ Jacobian-free Jacobian $\times$ matrix products)
- adjoints $\bar{F}_i$ ($\rightarrow$ Jacobian-free matrix $\times$ Jacobian products)

Wanted:

A feasible (persistent memory requirement) and (near-)optimal AD mission prescribing preferred modes (tangent or adjoint) for each module.

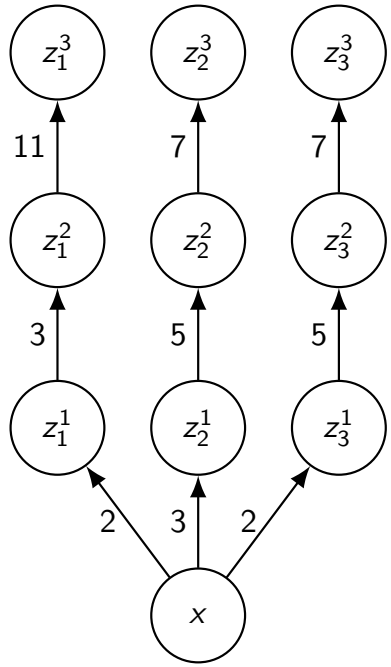
... yields the combinatorial AD Mission Planning problem.

AD Mission Planning is NP-hard.

Proof: The optimal evaluation of

$$\begin{pmatrix} 11 & & \\ & 7 & \\ & & 7 \end{pmatrix} \cdot \begin{pmatrix} 3 & & \\ & 5 & \\ & & 5 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix}$$

solves corresponding instance of the NP-complete Ensemble Computation (EC).



Trivial target graph structure in reduction from EC. Hardness due to algebraic dependences (e.g. equality) amongst local partial derivatives.

Revised reduction is valid for arbitrary order.

R. Garey and S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.
U.N.: Optimal Jacobian Accumulation is NP-Complete. Math. Prog. 112 (2), 2008.
U.N.: On Sparse Matrix Chain Products. SIAM CSC, 2020.
U.N.: On the Computational Complexity of the Chain Rule of Differential Calculus. arXiv:2107.05355 [cs.CC]. Submitted.

An optimal bracketing for $F'_p \cdot \ldots \cdot F'_1$ can be computed at the computational cost of $\mathcal{O}(p^3)$ by the recurrence

$$\textsc{Ert}_{k,i} = \begin{cases} 0 & k = i \\ \min_{i \le j < k} \left(\textsc{Ert}_{k,j+1} + \textsc{Ert}_{j,i} + \textsc{Ert}_{k,j,i} \right) & k > i \end{cases}$$

through tabulating $\textsc{Ert}_{k,i}$ for $k - i = 0, \ldots, p-1$ and where $\textsc{Ert}_{k,j,i}$ is the cost of evaluating $F'_{k,j+1} \cdot F'_{j,i}$ and $F'_{k,i} \equiv F'_k \cdot \ldots \cdot F'_i$.

An optimal bracketing for $F'_p \cdot \ldots \cdot F'_1$ can be computed at the computational cost of $\mathcal{O}(p^3)$ by the recurrence

$$\text{ERT}_{k,i} = \begin{cases} 0 & k = i \\ \min_{i \le j < k} \left(\text{ERT}_{k,j+1} + \text{ERT}_{j,i} + \text{ERT}_{k,j,i} \right) & k > i \end{cases}$$

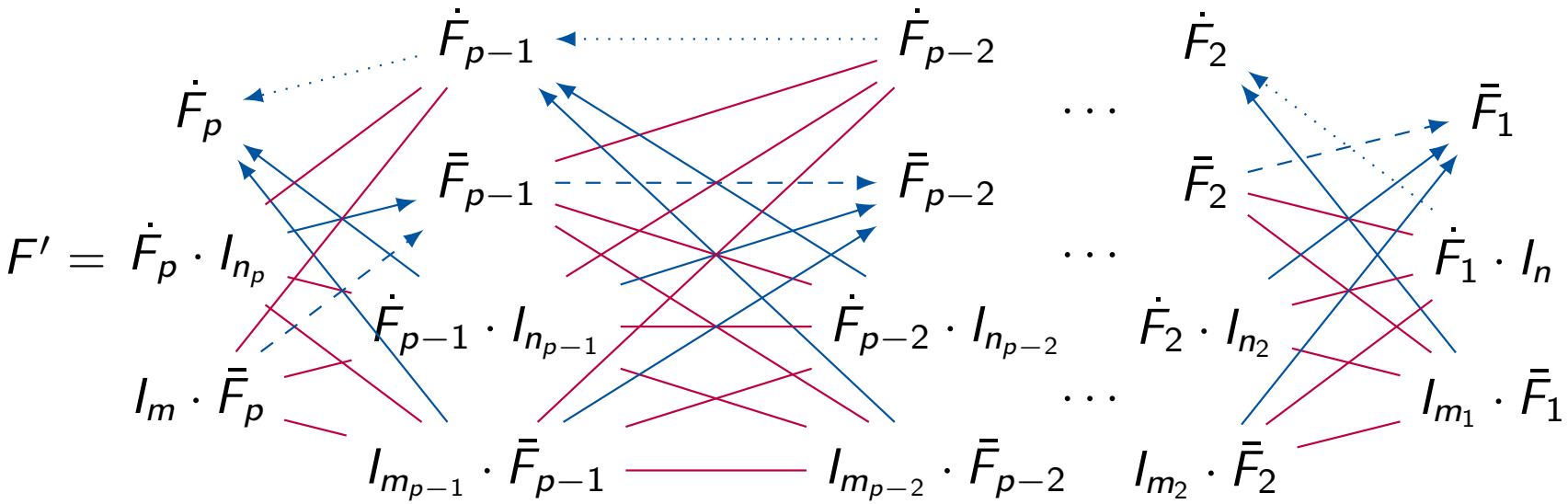
through tabulating $\text{ERT}_{k,i}$ for $k - i = 0, \ldots, p-1$ and where $\text{ERT}_{k,j,i}$ is the cost of evaluating $F'_{k,j+1} \cdot F'_{j,i}$ and $F'_{k,i} \equiv F'_k \cdot \ldots \cdot F'_i$.

Exploitation of sparsity increases the computational cost due to the need for explicit evaluation of the sparsity patterns for all subproblems. The total computational cost becomes equal to $\mathcal{O}(p^3 \cdot \nu^3)$ for $\nu = \max\left(n_1, \max_{\mu=1,\ldots,p} m_\mu\right)$.

Refer to the appendix for an example.

S. Godbole: On Efficient Computation of Matrix Chain Products. IEEE Trans. Comp., C-22(9), 1973.

We have tangents and adjoints – not Jacobians ...



... distinguishing standard and matrix-free matrix products. Note:

- feasible subproblems along directed paths (bidirectional purple edges)
- each layer visited once; $\dot{F}_1 \equiv \dot{F}_1 \cdot I_n$ and $\bar{F}_p \equiv I_m \cdot \bar{F}_p$
- homogeneous tangent dotted; homogeneous adjoint dashed;

A generalized [dense] Jacobian chain product of length two with $n = n_1 = 4$, $m_1 = n_2 = 2$, $m = m_2 = 32$, and $\text{ERT}(F_1) = \text{ERT}(F_2) = 100$ yields the following eight ($= 11 - 3$; refer to previous slide) bracketings:

- $F' = \dot{F}_2 \cdot F'_1 = \dot{F}_2 \cdot (\dot{F}_1 \cdot I_{n_1})$ $\Rightarrow \text{ERT}(F') = 800$
- $F' = \dot{F}_2 \cdot F'_1 = \dot{F}_2 \cdot (I_{m_1} \cdot \bar{F}_1)$ $\Rightarrow \text{ERT}(F') = 600$
- $F' = F'_2 \cdot \bar{F}_1 = (I_{m_2} \cdot \bar{F}_2) \cdot \bar{F}_1$ $\Rightarrow \text{ERT}(F') = 6400$
- $F' = F'_2 \cdot \bar{F}_1 = (\dot{F}_2 \cdot I_{n_2}) \cdot \bar{F}_1$ $\Rightarrow \text{ERT}(F') = 3400$
- $F' = F'_2 \cdot F'_1 = (\dot{F}_2 \cdot I_{n_2}) \cdot (I_{m_1} \cdot \bar{F}_1)$ $\Rightarrow \text{ERT}(F') = 656$
- $F' = F'_2 \cdot F'_1 = (I_{m_2} \cdot \bar{F}_2) \cdot (I_{m_1} \cdot \bar{F}_1)$ $\Rightarrow \text{ERT}(F') = 3656$
- $F' = F'_2 \cdot F'_1 = (I_{m_2} \cdot \bar{F}_2) \cdot (\dot{F}_1 \cdot I_{n_1})$ $\Rightarrow \text{ERT}(F') = 3856$
- $F' = F'_2 \cdot F'_1 = (\dot{F}_2 \cdot I_{n_2}) \cdot (\dot{F}_1 \cdot I_{n_1})$ $\Rightarrow \text{ERT}(F') = 856$

Optimal substructure and overlapping subproblems yield

$$\mathrm{ERT}_{j,i} = \begin{cases} \min\{n_j \cdot \mathrm{ERT}(\dot{F}), m_j \cdot \mathrm{ERT}(\bar{F})\} & j = i \\ \min_{i \le k < j} \left\{ \min \left\{ \begin{array}{l} \mathrm{ERT}_{j,k+1} + \mathrm{ERT}_{k,i} + m_j \cdot m_k \cdot n_i, \\ \mathrm{ERT}_{j,k+1} + m_j \cdot \sum_{\nu=i}^{k} \mathrm{ERT}(\bar{F}_\nu), \\ \mathrm{ERT}_{k,i} + n_i \cdot \sum_{\nu=k+1}^{j} \mathrm{ERT}(\dot{F}_\nu) \end{array} \right\} \right\} & j > i \,. \end{cases}$$

Note: The above assumes feasibility of the homogeneous adjoint.

Refer to the appendix for an example.

U.N.: Optimization of Generalized Jacobian Chain Products without Memory Constraints. arXiv:2003.05755 [math.NA]. Submitted.

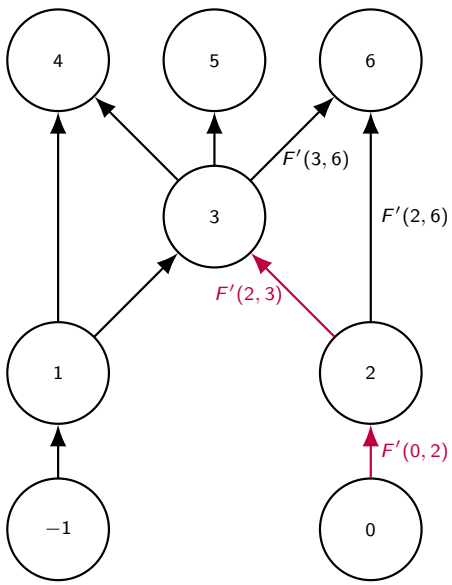
Numbers of fused multiply-adds for dense instances of random dimensions $\leq p$:

p	Tangent	Adjoint	Preaccumulation	Optimum
10	3,708	5,562	2,618	1,344
50	1,283,868	1,355,194	1,687,575	71,668
100	3,677,565	44,866,293	40,880,996	1,471,636
250	585,023,794	1,496,126,424	1,196,618,622	9,600,070
500	21,306,718,862	19,518,742,454	1,027,696,225	149,147,898

Refer to `https://github.com/un110076/ADMission.git` for the reference implementation.

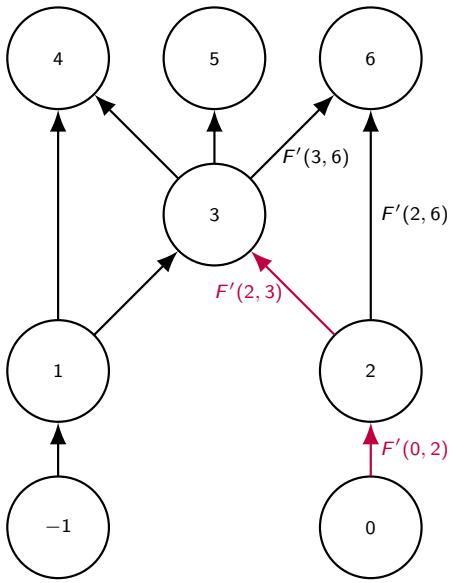
U.N.: Optimization of Generalized Jacobian Chain Products without Memory Constraints. arXiv:2003.05755 [math.NA]. Submitted.

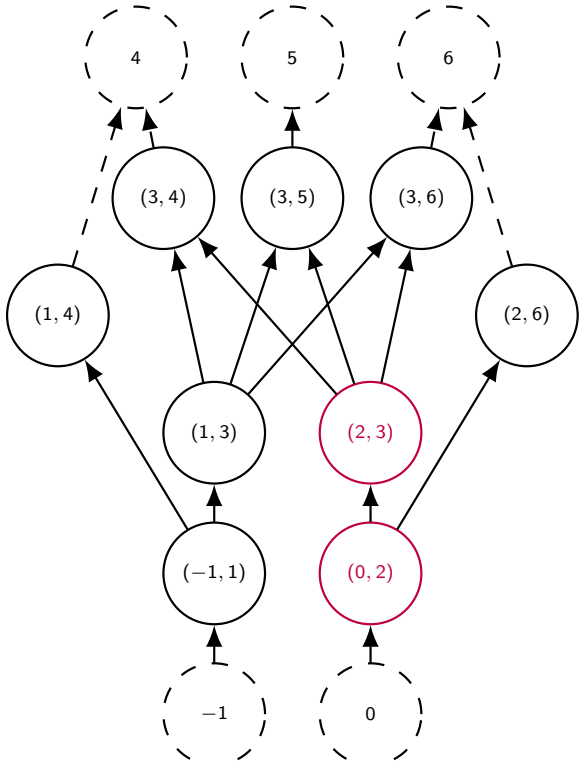
DAG G Face DAG $\tilde{G}$ $\tilde{G} - (0, 2, 3)$



Generalized Face Elimination

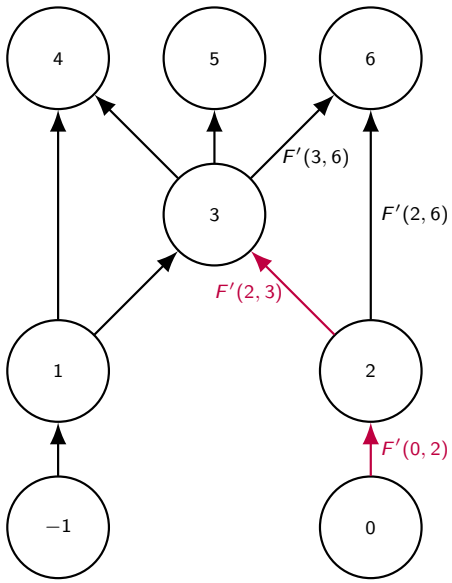
DAG G | Face DAG $\tilde{G}$ | $\tilde{G} - (0, 2, 3)$



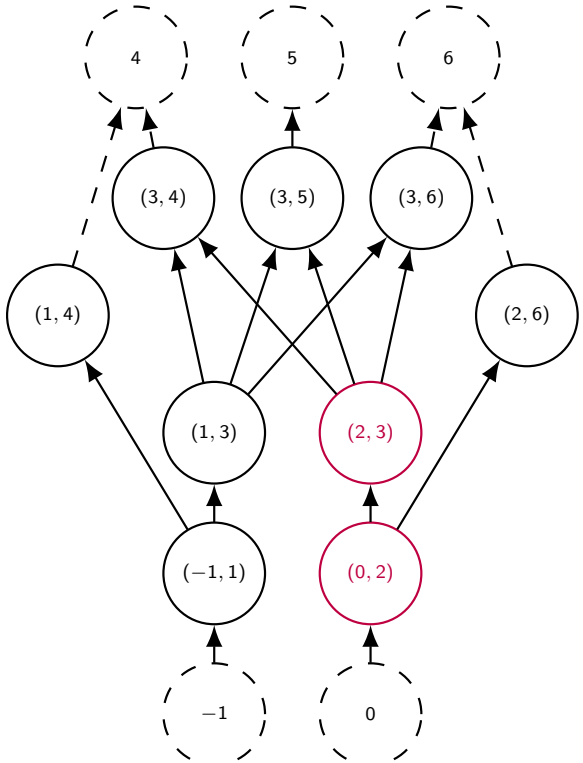


Generalized Face Elimination

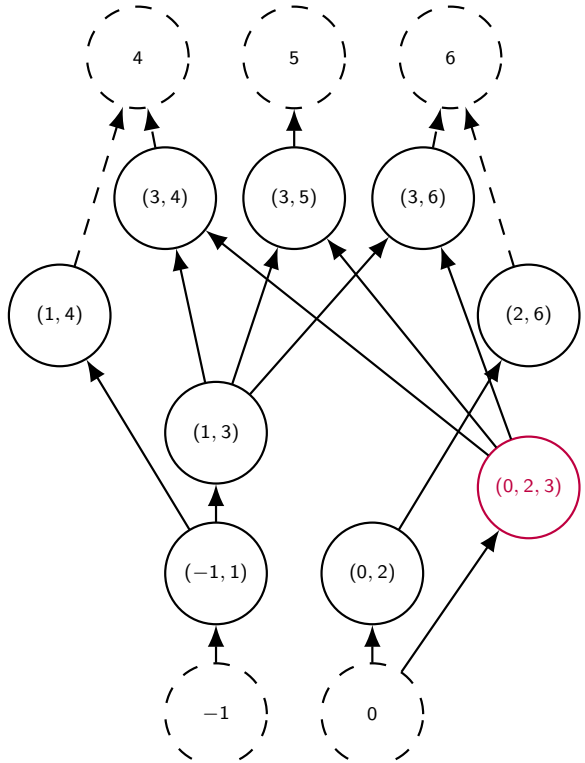
DAG G



Face DAG $\tilde{G}$



$\tilde{G} - (0, 2, 3)$



Work in progress: ADMISSION software

and corresponding combinatorial optimization problems ...

- Path Elimination

  W. Baur, V. Strassen: *The Complexity of Partial Derivatives.* Theor. Comp. Sci., 1983.

- Vertex Elimination

  A. Griewank, S. Reese: *On the Calculation of Jacobian Matrices by the Markowitz Rule.* AD @ Breckenridge, 1991.

- Edge Elimination

  U. N.: *Elimination Techniques for Cheap Jacobians.* AD @ Nice, 2000.

- Face Elimination

  U. N.: *Optimal Accumulation of Jacobian Matrices by Elimination Methods on the Dual Computational Graph.* Math. Prog., 2004.

- Generalized Face (Vertex, Edge) Elimination

  U. N. and E. Schneidereit: *Generalized Face Elimination in Computational Graphs.* Submitted.

Outline

Further Consequences

- Differential invariants

$$\bar{x} \cdot \dot{x} = (\bar{y} \cdot F'(x)) \cdot \dot{x} = \bar{y} \cdot (F'(x) \cdot \dot{x}) = \bar{y} \cdot \dot{y}$$

U. N.: *Differential Invariants.* arXiv:2101.03334 [math.NA]. Submitted.

- Derivative code design patterns: Early backpropagation / preaccumulation, Late recording / preaccumulation

U. N.: *Adjoint Code Design Patterns.* ACM TOMS 45(3):1-32, 2019.

- Hessian chain bracketing ($F = F_q \circ F_{q-1} \circ F_{q-2} \circ \cdots \circ F_1$)

$$[F'']_{\delta,\alpha_1,\alpha_2} = \sum_{j=1}^{q} \left[\prod_{i=j+1}^{q} F_i'\right]_{\delta,\gamma} \cdot \left[F_j''\right]_{\gamma,\beta_1,\beta_2} \cdot \left[\prod_{k=1}^{j-1} F_k'\right]_{\beta_1,\alpha_1} \cdot \left[\prod_{k=1}^{j-1} F_k'\right]_{\beta_2,\alpha_2}$$

U. N. and S. Burela: *Hessian Chain Bracketing.* arXiv:2103.09480 [cs.MS]. Submitted.

Outline

Conclusion

- The best choice out of homogeneous tangent and adjoint AD does not guarantee near-optimal runtime performance.
- Black-box adjoint AD works only in computationally inexpensive (serial run time of a few seconds) scenarios due to excessive persistent memory requirement.
- AD Mission Planning is NP-hard.
- (Generalized) Jacobian Chain Bracketing can be solved efficiently.
- (Generalized) Face Elimination remains NP-hard.
- AD mission planning is expected to benefit from ADMission software.

Outline

Other Recent Topics

- **Deterministic global optimization**

  J. Deussen and U. N.: *Subdomain separability in global optimization.* arXiv:2010.09591v1 [math.OC]. To appear in J. Glob. Opt.

- **Pruning of artificial neural networks**

  S. Afghan and U. N.: *Interval adjoint significance analysis for neural networks.* ICCS 2020, pages 365–378.

- **Algorithmic adjoint OpenFOAM**

  M. Towara, J. Lotz and U. N.: *Discrete adjoint approaches for CHT applications in OpenFOAM.* In Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences, pages 163–178, 2021.

- **Algorithmic differentiation of shock scenarios in CFD**

  M. Herty, J. Hüser, U. N. et al.: *Algorithmic differentiation of hyperbolic flow problems.* J Comp. Phys. 430, 110110, 2021.

- **Numerical stability of AD and the Implicit Function Theorem**

  U. N.: *Numerical Stability of Tangents and Adjoints of Implicit Functions.* ICCS 2022, pages 181–187.

- **Differential algebraic equations with optimality constraints**

  T. Ploch, J. Deussen, U. N. et al.: *Direct single shooting for dynamic optimization of differential-algebraic equation systems with optimization criteria embedded.* Comp. & Chem. Eng. 159, 107643, 2022.

Outline

Tangents with dco/c++

Let the primal f(ncp,ncs,eps,x,p,dW) solve a stochastic differential equation using the Euler-Maruyama / Monte-Carlo scheme.

```
1  #include "f.hpp" // primal source
2  #include "dco.hpp" // dco/c++
3
4  template<typename T, typename PT> // gradient driver
5  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) {
6    using DCO_T=typename dco::gt1s<T>::type; // scalar tangent type
7    size_t n=p_v.size(); // size of gradient
8    DCO_T x; std::vector<DCO_T> p(n); dco::value(p)=p_v; // activate
9    for (size_t i=0;i<n;i++) { // iterate over Cartesian basis
10     x=x_v; // [re]set inoutput
11     dco::derivative(p[i])=1; // seed
12     f(ncp,ncs,eps,x,p,dW); // compute primal
13     dxdp[i]=dco::derivative(x); // harvest
14     dco::derivative(p[i])=0; // unseed read-only input
15   }
16   x_v=dco::value(x); // get primal function value
17 }
```

Adjoints with dco/c++

```
1  #include "f.hpp" // primal source
2  #include "dco.hpp" // dco/c++
3
4  template<typename T, typename PT> // adjoint driver
5  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) {
6    using DCO_M=typename dco::ga1s<T>; // adjoint mode
7    using DCO_T=typename DCO_M::type; // adjoint type
8    using DCO_TT=typename DCO_M::tape_t; // tape type
9    size_t n=p_v.size(); // size of gradient
10   DCO_T x=x_v; std::vector<DCO_T> p(n); dco::value(p)=p_v; // activate
11   DCO_M::global_tape=DCO_TT::create(); // "touch" tape
12   DCO_M::global_tape->register_variable(p); // record active inputs
13   f(ncp,ncs,eps,x,p,dW); // compute augmented primal
14   x_v=dco::value(x); // get primal function value
15   DCO_M::global_tape->register_output_variable(x); // record active output
16   dco::derivative(x)=1; // seed
17   DCO_M::global_tape->interpret_adjoint(); // interpret tape
18   for (size_t i=0;i<n;i++) dxdp[i]=dco::derivative(p[i]); // harvest
19   DCO_TT::remove(DCO_M::global_tape); // deallocate tape
20 }
```

Bracketing of Jacobian Chain Products (Example)

If $F' = F'_4 \cdot F'_3 \cdot F'_2 \cdot F'_1$ with dense $F'_4 \in \mathbb{R}^{4\times 4}$, $F'_3 \in \mathbb{R}^{4\times 2}$, $F'_2 \in \mathbb{R}^{2\times 3}$, $F'_1 \in \mathbb{R}^{3\times 1}$, then

- $k - i = 0$:
  - $\text{ERT}_{i,i} = 0$ for $i = 4, \ldots, 1$
- $k - i = 1$:
  - $\text{ERT}_{4,3} = 4 \cdot 4 \cdot 2 = 32$ and $F'_{4,3} \in \mathbb{R}^{4\times 2}$
  - $\text{ERT}_{3,2} = 4 \cdot 2 \cdot 3 = 24$ and $F'_{3,2} \in \mathbb{R}^{4\times 3}$
  - $\text{ERT}_{2,1} = 2 \cdot 3 \cdot 1 = 6$ and $F'_{2,1} \in \mathbb{R}^{2\times 1}$
- $k - i = 2$:
  - $\text{ERT}_{4,2} = \min\{\text{ERT}_{3,2} + 4 \cdot 4 \cdot 3, \text{ERT}_{4,3} + 4 \cdot 2 \cdot 3\} = 56$ and $F'_{(4,3),2} \in \mathbb{R}^{4\times 3}$
  - $\text{ERT}_{3,1} = \min\{\text{ERT}_{2,1} + 4 \cdot 2 \cdot 1, \text{ERT}_{3,2} + 4 \cdot 3 \cdot 1\} = 14$ and $F'_{3,(2,1)} \in \mathbb{R}^{4\times 1}$
- $k - i = 3 = p - 1$:
  - $\text{ERT}_{4,1} = \min\{\text{ERT}_{3,1} + 4 \cdot 4 \cdot 1, \text{ERT}_{4,3} + \text{ERT}_{2,1} + 4 \cdot 2 \cdot 1, \text{ERT}_{4,2} + 4 \cdot 3 \cdot 1\} = 30$ and $F'_{4,(3,(2,1))} \in \mathbb{R}^{4\times 1}$

The optimal bracketing scheme $F'_4 \cdot (F'_3 \cdot (F'_2 \cdot F'_1))$ yields $\text{ERT}(F) = 30$ assuming unit cost per fused multiply-add.

Bracketing of Generalized Jacobian Chain Products (Example) I

Consider an instance of length $p = 3$ with

1. $n_1 = 3$, $m_1 = 3$, $\text{ERT}(\dot{F}_1) = \text{ERT}(\bar{F}_1) = 29$
2. $n_2 = 3$, $m_2 = 1$, $\text{ERT}(\dot{F}_2) = \text{ERT}(\bar{F}_2) = 14$
3. $n_3 = 1$, $m_3 = 2$, $\text{ERT}(\dot{F}_3) = \text{ERT}(\bar{F}_3) = 7$

Optimal preaccumulation of the local Jacobians induces the following costs:

1. F'_1 : $\text{ERT}_{1,1} = 87$
2. F'_2 : $\text{ERT}_{2,2} = 14$
3. F'_3 : $\text{ERT}_{3,3} = 7$

Two subproblems of length two need to be considered:

- $F'_{3,2} \equiv F'_3 \cdot F'_2$ comprising
  - $F'_3 \cdot F'_2$ at $\text{ERT}_{3,2} = 7 + 14 + 3 \cdot 1 \cdot 2 = 27$
  - $F'_3 \cdot \bar{F}_2$ at $\text{ERT}_{3,2} = 7 + 2 \cdot 14 = 35$
  - $\dot{F}_3 \cdot F'_2$ at $\text{ERT}_{3,2} = 14 + 3 \cdot 7 = 35$

  and, hence, yielding $\text{ERT}_{3,2} = 27$.
- $F'_{2,1} \equiv F'_2 \cdot F'_1$ comprising

Bracketing of Generalized Jacobian Chain Products (Example) II

- $F_2' \cdot F_1'$ at $\mathrm{ERT}_{2,1} = 87 + 14 + 3 \cdot 3 \cdot 1 = 110$
- $F_2' \cdot \bar{F}_1$ at $\mathrm{ERT}_{2,1} = 14 + 1 \cdot 29 = 43$
- $\dot{F}_2 \cdot F_1'$ at $\mathrm{ERT}_{2,1} = 87 + 3 \cdot 14 = 129$

and, hence, yielding $\mathrm{ERT}_{3,2} = 43$.

The last iteration compares

- $F_{3,2}' \cdot F_1'$ at $\mathrm{ERT}_{3,1} = 27 + 87 + 2 \cdot 3 \cdot 3 = 132$
- $F_{3,2}' \cdot \bar{F}_1$ at $\mathrm{ERT}_{3,1} = 27 + 2 \cdot 29 = 85$
- $\dot{F}_{3,2} \cdot F_1'$ at $\mathrm{ERT}_{3,1} = 87 + 3 \cdot (7 + 14) = 150$
- $F_3' \cdot F_{2,1}'$ at $\mathrm{ERT}_{3,1} = 7 + 43 + 2 \cdot 1 \cdot 3 = 56$
- $F_3' \cdot \bar{F}_{2,1}$ at $\mathrm{ERT}_{3,1} = 7 + 2 \cdot (14 + 29) = 93$
- $\dot{F}_3 \cdot F_{2,1}'$ at $\mathrm{ERT}_{3,1} = 43 + 3 \cdot 7 = 64$

yielding the optimal bracketing

$$(\dot{F}_3 \cdot I_{n_3}) \cdot ((I_{m_2} \cdot \bar{F}_2) \cdot \bar{F}_1)$$

with cost $\mathrm{ERT}_{3,1} = 56$.