Perturbation Confusion, Nesting, and The Amazing Bug

Oleksandr Manzyuk\textsuperscript{2}  \hspace{1cm} \textbf{Barak A. Pearlmutter}\textsuperscript{1,2}  \hspace{1cm} Alexey Andreyevich Radul\textsuperscript{2}

David R. Rush\textsuperscript{1}  \hspace{1cm} Jeffrey Mark Siskind\textsuperscript{3}

\textsuperscript{1}Dept Computer Science & \textsuperscript{2}Hamilton Institute, NUI Maynooth  
Co. Kildare, Ireland

\textsuperscript{3}School of Electrical and Computer Engineering, Purdue University  
West Lafayette IN 47907-2035, USA
Derivative-Taking Operator Desiderata

- functions whose domain and/or range can be
  - aggregates (e.g., gradients, div, curl, Jacobians, tuples, structures)
  - functions (e.g., filters, Fourier transforms, Laplace transforms, convolutions, Hamiltonians)
- that nest
  - higher-order derivatives (e.g., Hessians)
  - minimax
  - used ubiquitously
High-Level Summary

- Want to nest derivatives as confidently as we nest if statements.
- Derivatives taken by threading extra stuff through computation.
- Each nested derivative gets its own copy of threading machinery.
High-Level Summary

- Want to nest derivatives as confidently as we nest if statements.
- Derivatives taken by threading extra stuff through computation.
- Each nested derivative gets its own copy of the threading machinery.
- 1-to-1: invoke derivative operator ↔ take derivative.
High-Level Summary

▶ Want to nest derivatives as confidently as we nest if statements.
▶ Derivatives taken by threading extra stuff through computation.
▶ Each nested derivative gets its own copy of threading machinery.

▶ 1-to-1: invoke derivative operator $\leftrightarrow$ take derivative.

▶ Want to generalize: allow derivatives of higher-order functions.
▶ No longer 1-to-1: invoke derivative operator $\leftrightarrow$ take derivative!
▶ Confuses the threading machinery.
Things to Unpack

1. Automatic Differentiation
   Forward AD
   (we’ll do Forward AD; same issues in Reverse AD)

2. Threading machinery
   Nesting and Perturbation Confusion

3. Derivatives of Higher-Order Functions
   Higher-Order AD

4. Decoupling derivative taking & invocations
   The Amazing Bug
Augment numeric program:
original calculation & compute directional derivative.

Attach derivative to each number. Propagate.

Implementation: Nonstandard Interpretation

Replace $x : \mathbb{R}$ by pair, $x + x'\epsilon$, where $x' : \mathbb{R}$.
(Dual number: like $\mathbb{C}$ except $\epsilon^2 = 0$ but $\epsilon \neq 0$; Clifford, 1873)

Propagate according to calculus.
Lift basis functions: $f(x + x'\epsilon) \triangleq f(x) + f'(x)x'\epsilon$
where $f'(x)$ is the derivative of $f$ at $x$.
If $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ then $f'(x)$ is $m \times n$ Jacobian matrix.

Extract $\epsilon$-coefficient of output.
(1/4) Forward AD — Example

\[ D f x \triangleq \text{coefficient of } \epsilon \text{ in } f \ (x + 1\epsilon) = \text{tg} \ \epsilon \ (f \ (x + 1\epsilon)) \]

\[ f \ t \triangleq 1/(1 + \exp(-t)) \]

\[ f \ 2 = 1/(1 + \exp(-2)) = 1/(1 + 0.135) = 1/1.1353 = 0.881 \]

\[ D f \ 2 = \text{tg} \ \epsilon \ (f \ (2 + 1\epsilon)) = \text{tg} \ \epsilon \ (1/(1 + \exp(-(2 + 1\epsilon)))) \]
\[ = \text{tg} \ \epsilon \ (1/(1 + \exp(-2 - 1\epsilon))) \]
\[ = \text{tg} \ \epsilon \ (1/(1 + (0.135 + 0.135\epsilon))) \]
\[ = \text{tg} \ \epsilon \ (1/(1 + (0.135 + 0.135\epsilon))) \]
\[ = \text{tg} \ \epsilon \ (1/(1.135 + 0.135\epsilon)) \]
\[ = \text{tg} \ \epsilon \ (0.881 + 0.105\epsilon) \]
\[ = 0.105 \]
\( \mathcal{D} f \ x \triangleq \text{coefficient of } \epsilon \text{ in } f \ (x + 1\epsilon) = \text{tg} \ \epsilon \ (f \ (x + 1\epsilon)) \)

\( f \ t \triangleq 1/(1 + \exp(-t)) \)

\( f \ 2 = 1/(1 + \exp(-2)) = 1/(1 + 0.135) = 1/1.1353 = 0.881 \)

\( \mathcal{D} f \ 2 = \text{tg} \ \epsilon \ (f \ (2 + 1\epsilon)) = \text{tg} \ \epsilon \ (1/(1 + \exp(-(2 + 1\epsilon)))) \)

\( = \text{tg} \ \epsilon \ (1/(1 + \exp(-2 - 1\epsilon))) \)

\( = \text{tg} \ \epsilon \ (1/(1 + (0.135 + 0.135\epsilon))) \)

\( = \text{tg} \ \epsilon \ (1/(1 + (0.135 + 0.135\epsilon))) \)

\( = \text{tg} \ \epsilon \ (1/(1.135 + 0.135\epsilon)) \)

\( = \text{tg} \ \epsilon \ (0.881 + 0.105\epsilon) \)

\( = 0.105 \)
Enormous user demand for efficient nested derivatives from machine learning, graphics, computer vision, ...

- Learning to learn
- *Neural Radiance Fields (NeRFs)*
- *Neural ODEs*
- Interacting intelligent agents with mutual models
- Optimization using stochastic Newton’s methods
- (This list could go on for hours.)
Nesting Derivatives — Perturbation Confusion

... $D (\lambda x . ... D (\lambda y . g x y) ...)$ ...
... $\mathcal{D} (\lambda x \ldots \mathcal{D} (\lambda y \cdot g \, x \, y) \ldots)$ ...

\[
\frac{d}{dx} \left( x \left( \frac{d}{dy} \left|_{y=2} \right. \right) \right) \bigg|_{x=1} = \mathcal{D} (\lambda x \cdot x \cdot (\mathcal{D} (\lambda y \cdot x \cdot y) \, 2)) \, 1
\]
Nesting Derivatives — Perturbation Confusion

\[ \frac{d}{dx} \left( x \left( \frac{d}{dy} xy \bigg|_{y=2} \right) \right) \bigg|_{x=1} = \mathcal{D} (\lambda x \cdot x \cdot (\mathcal{D} (\lambda y \cdot x \cdot y) 2)) \]

**Key issue:** Need to distinguish the (co)tangents of \( x \) and \( y \).

*Each invocation of \( \mathcal{D} \) needs its own \( \epsilon \); note that \( \epsilon_1 \epsilon_2 \neq 0 \).*

\[ \mathcal{D} f x \equiv \text{fresh } \epsilon \text{ in } \text{tg } \epsilon \ (f \ (x + \epsilon)) \]

If you don’t, you get **perturbation confusion** and the **wrong answer** (Siskind and Pearlmutter, 2008).
Nesting Derivatives — Perturbation Confusion

\[ \ldots D (\lambda x \ldots D (\lambda y . g x y) \ldots) \ldots \]

\[ \frac{d}{dx} \left( x \left( \frac{d}{dy} x y \bigg|_{y=2} \right) \right) \bigg|_{x=1} = D (\lambda x . x \cdot (D (\lambda y . x \cdot y) 2)) 1 \]

Key issue: Need to distinguish the (co)tangents of \( x \) and \( y \).
Each invocation of \( D \) needs its own \( \epsilon \); note that \( \epsilon_1 \epsilon_2 \neq 0 \).

\[ D f x \triangleq \text{fresh } \epsilon \text{ in } tg \epsilon (f (x + \epsilon)) \]

If you don’t, you get perturbation confusion and the wrong answer (Siskind and Pearlmutter, 2008).

Recent formulations still get it wrong. It is out of scope for some formulations (some reject, some crash). Some get the wrong answer.
Nesting Derivatives — Perturbation Confusion

...$\mathcal{D} (\lambda x \ldots \mathcal{D} (\lambda y \cdot g x y) \ldots)\ldots$

\[
\frac{d}{dx} \left( x \left( \frac{d}{dy} xy \bigg|_{y=2} \right) \right) \bigg|_{x=1} = \mathcal{D} (\lambda x \cdot (\mathcal{D} (\lambda y \cdot x \cdot y) 2)) 1
\]

**Key issue:** Need to distinguish the (co)tangents of $x$ and $y$. 
*Each invocation of $\mathcal{D}$ needs its own $\epsilon$; note that $\epsilon_1 \epsilon_2 \neq 0$.*

\[
\mathcal{D} f x \triangleq \textbf{fresh } \epsilon \textbf{ in } \textbf{tg } \epsilon (f (x + \epsilon))
\]

If you don’t, you get **perturbation confusion** and the **wrong answer** (Siskind and Pearlmutter, 2008).

Recent formulations **still** get it wrong. It is out of scope for some formulations (some reject, some crash). Some get the **wrong answer**.

Most lack the machinery to distinguish (co)tangents.
Possible Definitions:

- Taking derivatives of higher-order functions
- Taking higher-order derivatives, like \( f''(x) \), of first-order functions
- Taking derivatives of first-order functions written in higher-order languages
Possible Definitions:

- Taking derivatives of higher-order functions
- Taking higher-order derivatives, like $f''(x)$, of first-order functions
- Taking derivatives of first-order functions written in higher-order languages

Many “higher-order AD” formulations use the other (easier) definitions.

What is meant by the derivative of a higher-order function?
Motivation for development of *Differential Geometry* in the 1800s.
Consider a simple higher-order function $f$: a curried function.

The derivative ($Df$) is the partial derivative WRT $f$’s first argument.

**Example:**

\[
\begin{align*}
  f(x, y) &\triangleq x^2 + y^3 + \pi \cdot x \cdot y \\
  f &\triangleq \lambda x \cdot \lambda y \cdot x^2 + y^3 + \pi \cdot x \cdot y \\
  f(5) &\triangleq \lambda y \cdot 25 + y^3 + 5 \cdot \pi \cdot y \\
  Df(5) &\triangleq \lambda y \cdot 10 + \pi \cdot y
\end{align*}
\]

This ignores the case of taking the derivative of a function whose *domain* is a function. See Manzyuk et al. (2019) for a discussion of that.
Higher-Order AD — Definitions

Derivative operator (calculates the derivative at a point):

\[ D : (\mathbb{R} \to \alpha) \to \mathbb{R} \to \alpha \]

\[ D f \ x \ \overset{\Delta}{=} \textbf{fresh} \ \varepsilon \ \text{in} \ \text{tg} \ \varepsilon \ (f \ (x + \varepsilon)) \]

Tangents are found by \( \text{tg} \), directly from numbers:

\[ \text{tg} \ \varepsilon \ (a + b\varepsilon) \ \overset{\Delta}{=} b \]

and by post-composition from functions:

\[ \text{tg} \ \varepsilon \ (\lambda x . \ e) \ \overset{\Delta}{=} (\text{tg} \ \varepsilon) \circ (\lambda x . \ e) \]
The Amazing Bug — Setup

Define offset operator:

$$s \ u \ f \ x \triangleq f(x + u)$$

and define:

$$\hat{D} \triangleq D \ s \ 0$$

Should have $$\hat{D} = D$$ since:

$$D \ f \ x = \text{fresh } \varepsilon \ \text{in } \operatorname{tg} \ \varepsilon (f(x + \varepsilon)) = \operatorname{tg} \ \varepsilon (f(x) + f'(x)\varepsilon) = f'(x)$$

$$\hat{D} \ f \ x = D \ s \ 0 \ f \ x = (\frac{d}{du}) s \ u \ f \ x \bigg|_{u=0}$$

$$= (\frac{d}{du}) f(x + u) \bigg|_{u=0} = f'(x)$$
The Amazing Bug — Manifestation

Should have $\hat{\mathcal{D}} = \mathcal{D}$, but

$$\mathcal{D} (\mathcal{D} f) \, x = f'''(x)$$

$$\hat{\mathcal{D}} (\hat{\mathcal{D}} f) \, x = 0$$
Recall:

\[ \hat{D} = D \ s \ 0 = \textbf{fresh} \ e \ \textbf{in} \ \textbf{tg} \ e \ (s \ (0 + e)) = \textbf{tg} \ e \ (\lambda f . \lambda x . f \ (x + e)) \]

\[ = \lambda f . \lambda x . \textbf{tg} \ e \ (f \ (x + e)) \]

Assume \( h : \mathbb{R} \rightarrow \mathbb{R} \), substitute and reduce:

\[ \hat{D} \ (\hat{D} \ h) \ y = (\lambda f . \lambda x . \textbf{tg} \ e \ (f \ (x + e))) \ ((\lambda f . \lambda x . \textbf{tg} \ e \ (f \ (x + e))) \ h) \ y \]

\[ = (\lambda f . \lambda x . \textbf{tg} \ e \ (f \ (x + e))) \ (\lambda x . \textbf{tg} \ e \ (h \ (x + e))) \ y \]

\[ = (\lambda x . \textbf{tg} \ e \ ((\lambda x . \textbf{tg} \ e \ (h \ (x + e))) \ (x + e))) \ y \]

\[ = \textbf{tg} \ e \ ((\lambda x . \textbf{tg} \ e \ (h \ (x + e))) \ (y + e)) \]

\[ = \textbf{tg} \ e \ (\textbf{tg} \ e \ (h \ ((y + e) + e))) \]

\[ = \textbf{tg} \ e \ (\textbf{tg} \ e \ (h \ (y + 2e))) \]

\[ = \textbf{tg} \ e \ (\textbf{tg} \ e \ (h(y) + 2h'(y)e)) \]

\[ = \textbf{tg} \ e \ (2h'(y)) \]

\[ = 0 \]
Recall:

\[
\hat{D} = D s 0 = \text{fresh } \varepsilon \text{ in } tg \varepsilon (s (0 + \varepsilon)) = tg \varepsilon (\lambda f . \lambda x . f (x + \varepsilon))
\]

\[
= \lambda f . \lambda x . \text{tg } \varepsilon (f (x + \varepsilon))
\]

Assume \( h : \mathbb{R} \rightarrow \mathbb{R} \), substitute and reduce:

\[
\hat{D} (\hat{D} h) y = (\lambda f . \lambda x . \text{tg } \varepsilon (f (x + \varepsilon))) ((\lambda f . \lambda x . \text{tg } \varepsilon (f (x + \varepsilon))) h) y
\]

\[
= (\lambda f . \lambda x . \text{tg } \varepsilon (f (x + \varepsilon))) (\lambda x . \text{tg } \varepsilon (h (x + \varepsilon))) y
\]

\[
= (\lambda x . \text{tg } \varepsilon ((\lambda x . \text{tg } \varepsilon (h (x + \varepsilon))) (x + \varepsilon))) y
\]

\[
= \text{tg } \varepsilon ((\lambda x . \text{tg } \varepsilon (h (x + \varepsilon))) (y + \varepsilon))
\]

\[
= \text{tg } \varepsilon (\text{tg } \varepsilon (h ((y + \varepsilon) + \varepsilon)))
\]

\[
= \text{tg } \varepsilon (\text{tg } \varepsilon (h (y + 2\varepsilon)))
\]

\[
= \text{tg } \varepsilon (\text{tg } \varepsilon (h(y) + 2h'(y)\varepsilon))
\]

\[
= \text{tg } \varepsilon (2h'(y))
\]

\[
= 0
\]
Okay, all unpacked!
Note: $\mathcal{D}$ was invoked once, in $\hat{\mathcal{D}} = \mathcal{D} s 0$. 
The Amazing Bug — Root Cause

Note: $\mathcal{D}$ was invoked once, in $\mathcal{D} = \mathcal{D} s 0$.

But we can still get nested derivatives.
The Amazing Bug — Root Cause

Note: $\mathcal{D}$ was invoked once, in $\hat{\mathcal{D}} = \mathcal{D} \ s \ 0$.

But we can still get nested derivatives.

This is because derivatives of higher-order functions breaks the 1-to-1 relationship between invoking a derivative-taking operator and taking a derivative.
The Amazing Bug — Root Cause

Note: $D$ was invoked once, in $\hat{D} = D \cdot s \cdot 0$.

But we can still get nested derivatives.

This is because derivatives of higher-order functions *breaks* the 1-to-1 relationship between invoking a derivative-taking operator and taking a derivative.

**Key issue:** Need to distinguish the (co)tangents of two different derivatives even though the derivative operator is called only once. If you don’t, you get *perturbation confusion* and the *wrong answer* *despite the anti-perturbation-confusion tagging machinery*. 
The Amazing Bug — A Workaround

Get correct result if $\hat{D} = D s 0$ is left un-reduced.

$$\hat{D} (\hat{D} g) y = D s 0 (D s 0 g) y$$

$$= (\lambda f . \lambda x . \text{tg } \epsilon_1 (f (x + \epsilon_1)))$$

$$((\lambda f . \lambda x . \text{tg } \epsilon_2 (f (x + \epsilon_2))) g) y$$

$$= (\lambda f . \lambda x . \text{tg } \epsilon_1 (f (x + \epsilon_1)))$$

$$\text{tg } \epsilon_2 (g (x + \epsilon_2)) y$$

$$= (\lambda x . \text{tg } \epsilon_1 ((\lambda x . \text{tg } \epsilon_2 (g (x + \epsilon_2))) (x + \epsilon_1))) y$$

$$= \epsilon_1 (((\lambda x . \text{tg } \epsilon_2 (g (x + \epsilon_2))) (y + \epsilon_1))$$

$$= \text{tg } \epsilon_1 (\text{tg } \epsilon_2 (g ((y + \epsilon_1) + \epsilon_2)))$$

$$= \text{tg } \epsilon_1 (\text{tg } \epsilon_2 (g (y + \epsilon_1) + g'(y + \epsilon_1)\epsilon_2))$$

$$= \text{tg } \epsilon_1 g'(y + \epsilon_1)$$

$$= \text{tg } \epsilon_1 (g' (y) + g''(y)\epsilon_1)$$

$$= g''(y)$$
The Essence of the Above Workaround

Write

\[
\begin{align*}
\textbf{let } & \quad \hat{D}_1 \triangleq \mathcal{D} \ s \ 0 \\
\hat{D}_2 & \triangleq \mathcal{D} \ s \ 0 \\
\textbf{in } & \quad \hat{D}_1 (\hat{D}_2 \ h) \ y
\end{align*}
\]

instead of

\[
\begin{align*}
\textbf{let } & \quad \hat{D} \triangleq \mathcal{D} \ s \ 0 \\
\textbf{in } & \quad \hat{D} (\hat{D} \ h) \ y
\end{align*}
\]
The Essence of the Above Workaround

Write

\[
\text{let } \hat{\mathcal{D}}_1 \triangleq \mathcal{D} \ s \ 0 \\
\hat{\mathcal{D}}_2 \triangleq \mathcal{D} \ s \ 0 \\
\text{in } \hat{\mathcal{D}}_1 (\hat{\mathcal{D}}_2 h) \ y
\]

instead of

\[
\text{let } \hat{\mathcal{D}} \triangleq \mathcal{D} \ s \ 0 \\
\text{in } \hat{\mathcal{D}} (\hat{\mathcal{D}} h) \ y
\]
Solution Idea One: Eta Expansion

Delay \textbf{fresh} until all args needed for post-composition of \( \text{tg} \) are available, so it immediately beta reduces to non-function-containing value.

\[
\mathcal{D}_1 : (\mathbb{R} \to \mathbb{R}) \to \mathbb{R} \to \mathbb{R}
\]

\[
\mathcal{D}_1 f \ x_1 \triangleq \textbf{fresh} \ \varepsilon \ \text{in} \ \text{tg} \ \varepsilon (f (x_1 + \varepsilon))
\]

\[
\mathcal{D}_2 : (\mathbb{R} \to \alpha_2 \to \mathbb{R}) \to \mathbb{R} \to \alpha_2 \to \mathbb{R}
\]

\[
\mathcal{D}_2 f \ x_1 \ x_2 \triangleq \textbf{fresh} \ \varepsilon \ \text{in} \ \text{tg} \ \varepsilon (f (x_1 + \varepsilon) \ x_2)
\]

\[\vdots\]

\[
\mathcal{D}_n : (\mathbb{R} \to \alpha_2 \to \cdots \to \alpha_n \to \mathbb{R}) \to \mathbb{R} \to \alpha_2 \to \cdots \to \alpha_n \to \mathbb{R}
\]

\[
\mathcal{D}_n f \ x_1 \cdots x_n \triangleq \textbf{fresh} \ \varepsilon \ \text{in} \ \text{tg} \ \varepsilon (f (x_1 + \varepsilon) \ x_2 \cdots x_n)
\]

\[\vdots\]
Solution Idea One: Eta Expansion

Or with polymorphic recursion:

\[\mathcal{D} f x \triangleq \lambda y . (\mathcal{D} (\lambda x . (f x y)) x)\]  
\[\mathcal{D} f x \triangleq \text{fresh } \varepsilon \text{ in } \text{tg } \varepsilon (f (x + \varepsilon))\]

\((f x)\) is a function

\((f x)\) is not a function

Can get hairy and dynamic when the type of \(f\) becomes more complicated.
Solution Idea Two: Tag Substitution

Guard Returned Function against Tag Collision

Instead of:

$$\text{tg } \varepsilon (\lambda x . \ e) \triangleq (\text{tg } \varepsilon) \circ (\lambda x . \ e)$$

augment the wrapper to guard the tag:

$$\text{tg } \varepsilon_1 (\lambda x . \ e) \triangleq \textbf{fresh } \varepsilon_2 \textbf{ in } [\varepsilon_1/\varepsilon_2] \circ (\text{tg } \varepsilon_1) \circ (\lambda x . \ e) \circ [\varepsilon_2/\varepsilon_1]$$

where $[\varepsilon_2/\varepsilon_1]$ substitutes $\varepsilon_2$ for every free occurrence of $\varepsilon_1$ in its argument, using pre- and post-composition on functions, also guarding against tag collision.
Conclusion

- Can import standard defs of higher-order function derivatives into AD.
- Allowing them breaks AD machinery.
- Two solution frameworks:
  1. eta expansion
  2. rename the (co)tangents captured in returned closures

Recent formulations still get it wrong. Some get wrong answer. One popular fielded formulation got the wrong answer for a quarter of a century. All lack the machinery to eta expand or rename captured (co)tangents.

Future work: efficiency and AD-complexity-guarantee issues remain. (See JFP paper, Manzyuk et al., 2019, for details.)
Conclusion

- Can import standard defs of higher-order function derivatives into AD.
- Allowing them breaks AD machinery.
- Two solution frameworks:
  1. eta expansion
  2. rename the (co)tangents captured in returned closures

Recent formulations *still* get it wrong.

Out of scope for some (reject or crash).

Some get wrong answer. (One popular fielded formulation got the wrong answer for a quarter of a century.) All lack the machinery to eta expand or rename captured (co)tangents.

Future work: efficiency and AD-complexity-guarantee issues remain.

(See JFP paper, Manzyuk et al., 2019, for details.)
ACKNOWLEDGEMENTS

This work was supported, in part, by Science Foundation Ireland (SFI) Principal Investigator grant 09/IN.1/I2637, by the Army Research Laboratory (ARL), accomplished under Cooperative Agreement Number W911NF-10-2-0060, by the National Science Foundation (NSF) under Grants 1522954-IIS and 1734938-IIS, and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00341. Any opinions, findings, views, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, official policies, or endorsements, either expressed or implied, of SFI, ARL, NSF, IARPA, DOI/IBC, or the Irish or U.S. Governments. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.
References I


