Automatic Differentiation of Functional Programs or Lambda the Ultimate Calculus

Jeffrey Mark Siskind
qobi@purdue.edu

School of Electrical and Computer Engineering
Purdue University

University of Chicago
8 April 2008

Joint work with Barak A. Pearlmutter.

Karl Gustav Jacob Jacobi

Friedrich Julius Richelot

Carl Gottfried Neumann

William Edward Story

Solomon Lefschetz

Albert William Tucker

Marvin Lee Minsky

Patrick Henry Winston	Scott Fahlman
Robert C. Berwick	David Touretzky
Jeffrey Mark Siskind	Barak Avrum Pearlmutter

It is, of course, not excluded that the range of arguments or range of values of a function should consist wholly or partly of functions. The derivative, as this notion appears in the elementary differential calculus, is a familiar mathematical example of a function for which both ranges consist of functions.

(¶4)

Church, A. (1941). *The Calculi of Lambda Conversion*, Princeton University Press, Princeton, NJ.

Gottfried Leibniz
|
Jacob Bernoulli
|
Johann Bernoulli
|
Leonhard Euler
|
Joseph Louis Lagrange
|
Simeon Poisson
|
Michel Chasles
|
Hubert Anson Newton
|
Eliakim Hastings Moore
|
Oswald Veblen
|
Alonzo Church

Leibnitz (1664) + Church (1941) = Siskind & Pearlmutter (2008)

Leibnitz, G. W. (1664). A new method for maxima and minima as well as tangents, which is impeded neither by fractional nor irrational quantities, and a remarkable type of calculus for this, *Acta Eruditorum*.

Higher-order functions are common in mathematics, physics, and engineering:

> *derivatives, gradients, Jacobians, summations, comprehensions, quantifications, optimizations, integrals, convolutions, filters, edge detectors, Fourier transforms, differential equations, Hamiltonians,* ...

where they are traditionally called *operators*.

Automatic Differentiation (AD)

$$3x^2 \longrightarrow \boxed{\frac{\mathrm{d}}{\mathrm{d}x}} \longrightarrow 6x$$

Automatic Differentiation (AD)

$$3x^2 \longrightarrow \boxed{\frac{\mathrm{d}}{\mathrm{d}x}} \longrightarrow 6x$$

```
function f(x)
f = 3*x**2
end
```

$\longrightarrow$ ADIFOR $\longrightarrow$

```
function fprime(x)
fprime = 6*x
end
```


Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Beda, L. M. *et al.* (1959). *Programs for Automatic Differentiation for the Machine BESM*, Inst. for Precise Mechanics and Computation Techniques, Academy of Science, Moscow

Finite Differences

$$\frac{f(x+\Delta x)-f(x)}{\Delta x}$$

A Xillion Implementations of AD

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)

FORTRAN: ADIFOR (Bischof *et al.*, 1996)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)
C: ADIC (Bischof *et al.*, 1997)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)
C: ADIC (Bischof *et al.*, 1997)
HASKELL: (Karczmarczuk, 1998, 1999, 2001; Nilsson, 2003)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)
C: ADIC (Bischof *et al.*, 1997)
HASKELL: (Karczmarczuk, 1998, 1999, 2001; Nilsson, 2003)
SCHEME: SCMUTILS (Sussman *et al.*, 2001)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)
C: ADIC (Bischof *et al.*, 1997)
HASKELL: (Karczmarczuk, 1998, 1999, 2001; Nilsson, 2003)
SCHEME: SCMUTILS (Sussman *et al.*, 2001)
MATLAB: ADIMAT (Bischof *et al.*, 2003)

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)
C: ADIC (Bischof *et al.*, 1997)
HASKELL: (Karczmarczuk, 1998, 1999, 2001; Nilsson, 2003)
SCHEME: SCMUTILS (Sussman *et al.*, 2001)
MATLAB: ADIMAT (Bischof *et al.*, 2003)
⋮

A Xillion Implementations of AD

MAPLE: GRADIENT (Monagan & Neuenschwander, 1993)
FORTRAN: ADIFOR (Bischof *et al.*, 1996)
C++: FADBAD++ (Bendtsen & Stauning, 1996)
C: ADIC (Bischof *et al.*, 1997)
HASKELL: (Karczmarczuk, 1998, 1999, 2001; Nilsson, 2003)
SCHEME: SCMUTILS (Sussman *et al.*, 2001)
MATLAB: ADIMAT (Bischof *et al.*, 2003)
⋮

`http://www.autodiff.org`

What's Novel?

What's Novel?

- AD for functional programs

Karczmarczuk, J. K. (2001). Functional differentiation of computer programs, *Higher Order and Symbolic Computation*, **14**:35–57.

What's Novel?

- AD for functional programs
- formulated as a higher-order function (in the language)

Karczmarczuk, J. K. (2001). Functional differentiation of computer programs, *Higher Order and Symbolic Computation*, **14**:35–57.

What's Novel?

- AD for functional programs
- formulated as a higher-order function (in the language)
- that reflectively transforms code and data (in closures)

Karczmarczuk, J. K. (2001). Functional differentiation of computer programs, *Higher Order and Symbolic Computation*, **14**:35–57.

What's Novel?

- AD for functional programs
- formulated as a higher-order function (in the language)
- that reflectively transforms code and data (in closures)
- in a way that exhibits closure properties

Karczmarczuk, J. K. (2001). Functional differentiation of computer programs, *Higher Order and Symbolic Computation*, **14**:35–57.

What's Novel?

- AD for functional programs
- formulated as a higher-order function (in the language)
- that reflectively transforms code and data (in closures)
- in a way that exhibits closure properties
- and a compiler that compiles away that reflection

Karczmarczuk, J. K. (2001). Functional differentiation of computer programs, *Higher Order and Symbolic Computation*, **14**:35–57.

Everything You Always Wanted to Know About the Lambda Calculus*

*But Were Afraid To Ask

Everything You Always Wanted to Know About the Lambda Calculus*

(in 8 slides)

*But Were Afraid To Ask

Functional Programming

```
int f(int n)
{ int i, p = 1;
  for (i = 1; i<n; i++)
  { p = p*i;}
  return p;}
```

Functional Programming

```
int f(int n)
{ int i, p = 1;
  for (i = 1; i<n; i++)
  { p = p*i;}
  return p;}
```

$$f\ n \stackrel{\triangle}{=} \textbf{if}\ n = 0\ \textbf{then}\ 1\ \textbf{else}\ n \times (f\ (n-1))$$

Higher-Order Functions

$$\sum_{i=1}^{n} \exp\ i \qquad\qquad \prod_{i=1}^{n} \sin\ i$$

Higher-Order Functions

$$\sum_{i=1}^{n} \exp\ i \qquad \prod_{i=1}^{n} \sin\ i$$

$$\begin{aligned}\textsc{Fold}\ i, a, f, g \triangleq\ & \textbf{if}\ i = 0\\ & \textbf{then}\ a\\ & \textbf{else}\ \textsc{Fold}\ (i-1), (g\ a, (f\ i)), f, g\end{aligned}$$

Higher-Order Functions

$$\sum_{i=1}^{n} \exp\ i \qquad\qquad \prod_{i=1}^{n} \sin\ i$$

$$\begin{aligned}\textsc{Fold}\ i, a, f, g \triangleq\ & \textbf{if}\ i = 0\\ & \textbf{then}\ a\\ & \textbf{else}\ \textsc{Fold}\ (i-1), (g\ a, (f\ i)), f, g\end{aligned}$$

$$\textsc{Fold}\ n, 0, \exp, + \qquad\qquad \textsc{Fold}\ n, 1, \sin, \times$$

$$\sum_{i=1}^{n} \exp\ i \qquad\qquad \prod_{i=1}^{n} \sin\ i$$

$$\begin{aligned}\textsc{Fold}\ i, a, f, g \triangleq\ & \textbf{if}\ i = 0\\ & \textbf{then}\ a\\ & \textbf{else}\ \textsc{Fold}\ (i-1), (g\ a, (f\ i)), f, g\end{aligned}$$

$$\textsc{Fold}\ n, 0, \exp, + \qquad\qquad \textsc{Fold}\ n, 1, \sin, \times$$

$$\sum_{i=1}^{n} 2i+1$$

Higher-Order Functions

$$\sum_{i=1}^{n} \exp\ i \qquad\qquad \prod_{i=1}^{n} \sin\ i$$

$$\begin{aligned}\textsc{Fold}\ i, a, f, g \stackrel{\triangle}{=}\ & \textbf{if}\ i = 0\\ & \textbf{then}\ a\\ & \textbf{else}\ \textsc{Fold}\ (i-1), (g\ a, (f\ i)), f, g\end{aligned}$$

$$\textsc{Fold}\ n, 0, \exp, + \qquad\qquad \textsc{Fold}\ n, 1, \sin, \times$$

$$\sum_{i=1}^{n} 2i+1$$

$$f\ i \stackrel{\triangle}{=} 2i+1 \qquad\qquad \textsc{Fold}\ n, 0, f, +$$

Higher-Order Functions

$$\sum_{i=1}^{n} \exp\ i \qquad\qquad \prod_{i=1}^{n} \sin\ i$$

$$\begin{aligned}\textsc{Fold}\ i, a, f, g \stackrel{\triangle}{=}\ & \textbf{if}\ i = 0\\ & \textbf{then}\ a\\ & \textbf{else}\ \textsc{Fold}\ (i-1), (g\ a, (f\ i)), f, g\end{aligned}$$

$$\textsc{Fold}\ n, 0, \exp, + \qquad\qquad \textsc{Fold}\ n, 1, \sin, \times$$

$$\sum_{i=1}^{n} 2i + 1$$

$$f\ i \stackrel{\triangle}{=} 2i + 1 \qquad\qquad \textsc{Fold}\ n, 0, f, +$$

$$\textsc{Fold}\ n, 0, (\lambda i\ 2i + 1), +$$

Closures

$$(\lambda x\ 2x)\ 3 = 6$$

Closures

$$(\lambda x\ 2x)\ 3 = 6$$

$$(\lambda x\ \lambda y\ x + y)\ 3\ 4 = 7$$

Closures

$$(\lambda x\ 2x)\ 3 = 6$$

$$(\lambda x\ \lambda y\ x + y)\ 3\ 4 = 7$$

$$(\lambda x\ \lambda y\ x + y)\ 3 = ?$$

Closures

$$(\lambda x\ 2x)\ 3 = 6$$

$$(\lambda x\ \lambda y\ x + y)\ 3\ 4 = 7$$

$$(\lambda x\ \lambda y\ x + y)\ 3 = \langle \{x \mapsto 3\}, \lambda y\ x + y \rangle$$

Closures

$$(\lambda x\ 2x)\ 3 = 6$$

$$(\lambda x\ \lambda y\ x + y)\ 3\ 4 = 7$$

$$(\lambda x\ \lambda y\ x + y)\ 3 = \langle \{x \mapsto 3\}, \lambda y\ x + y \rangle$$

$$\lambda x\ \lambda y\ x + y \qquad\qquad \lambda(x, y)\ x + y$$

Closure Conversion

$$\boxed{\begin{array}{l} f = \lambda y\, x + y \\ f\ 4 \end{array}}$$

Closure Conversion

$$\boxed{\begin{array}{l} f = \lambda y\, x + y \\ f\ 4 \end{array}} \quad \leadsto \quad \boxed{\begin{array}{l} f = (x, \lambda y\, x + y) \\ (\textsc{Cdr}\ f)\ ((\textsc{Car}\ f), 4) \end{array}}$$

Johnsson, T. (1985). Lambda Lifting: Transforming Programs to Recursive Equations, *Proceedings Functional Programming Languages and Computer Architecture*.

The Lambda Calculus

$$\textbf{if } e_1 \textbf{ then } e_2 \textbf{ else } e_3 \textbf{ fi} \quad \rightsquigarrow \quad \textsc{If}\ e_1\ (\lambda x\ e_2)\ (\lambda x\ e_3)\ [\,]$$

The Lambda Calculus

$$\textbf{if } e_1 \textbf{ then } e_2 \textbf{ else } e_3 \textbf{ fi} \quad \rightsquigarrow \quad \textsc{If}\ e_1\ (\lambda x\ e_2)\ (\lambda x\ e_3)\ [\,]$$

$$e ::= x \mid e_1\ e_2 \mid \lambda x\ e$$

A-Normal Form

$$\textbf{let } x = e_1 \textbf{ in } e_2 \qquad \rightsquigarrow \qquad (\lambda x\ e_2)\ e_1$$

A-Normal Form

$$
\begin{array}{lcl}
\textbf{let } x = e_1 \textbf{ in } e_2 & \rightsquigarrow & (\lambda x\ e_2)\ e_1 \\
\textbf{let } x_1 = e_1; & \rightsquigarrow & \textbf{let } x_1 = e_1 \\
\quad x_2 = e_2; & & \textbf{in let } x_2 = e_2; \\
\quad \vdots & & \quad\quad \vdots \\
\textbf{in } e & & \quad \textbf{in } e
\end{array}
$$

A-Normal Form

$$
\begin{array}{lcl}
\textbf{let}\ x = e_1\ \textbf{in}\ e_2 & \rightsquigarrow & (\lambda x\ e_2)\ e_1 \\
\textbf{let}\ x_1 = e_1; & \rightsquigarrow & \textbf{let}\ x_1 = e_1 \\
\qquad x_2 = e_2; & & \textbf{in let}\ x_2 = e_2; \\
\qquad \vdots & & \qquad \vdots \\
\textbf{in}\ e & & \qquad \textbf{in}\ e \\
(f_n\ \ldots (f_2\ (f_1\ x_0))\ldots) & \rightsquigarrow & \textbf{let}\ x_1 = f_1\ x_0; \\
& & \qquad x_2 = f_2\ x_1; \\
& & \qquad \vdots \\
& & \qquad x_n = f_n\ x_{n-1} \\
& & \textbf{in}\ x_n
\end{array}
$$

Sabry, A. and Felleisen, M. (1993). Reasoning about Programs in Continuation-Passing Style, *Lisp and Symbolic Computation*, **3**(3–4):289–360.

Nonstandard Interpretations

$$
\begin{array}{lcl}
3+4 & = & 7 \\
\boxed{\begin{array}{l}\textbf{let}\ + = - \\ \textbf{in}\ 3+4\end{array}} = -1 & &
\end{array}
$$

Monovariant Flow Analysis

needs work

Polyvariant Flow Analysis

needs work

Shivers, III, O. G. (1991). *Control-Flow Analysis of Higher-Order Languages* ***or*** *Taming Lambda*, Ph.D. thesis, CMU.

⋮

|

Alonzo Church

|

Stephen Cole Kleene

|

Robert Lee Constable

|

Steven Stanley Muchnick

|

Uwe Frederik Pleban

|

Peter Lee

|

Olin Shivers

Differential Calculus for Dummies

Differential Calculus for Dummies

(in 7 slides)

Derivatives

$$\frac{\mathrm{d}ax^2}{\mathrm{d}x} \rightsquigarrow 2ax$$

Derivatives

$$\frac{\mathrm{d}ax^2}{\mathrm{d}x} \rightsquigarrow 2ax$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : \underbrace{f}_{\mathbb{R}\to\mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}\to\mathbb{R}}$$

Derivatives

$$\frac{\mathrm{d}ax^2}{\mathrm{d}x} \rightsquigarrow 2ax$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : \underbrace{f}_{\mathbb{R}\to\mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}\to\mathbb{R}}$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

Derivatives

$$\frac{\mathrm{d}ax^2}{\mathrm{d}x} \rightsquigarrow 2ax$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : \underbrace{f}_{\mathbb{R}\to\mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}\to\mathbb{R}}$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

$$\mathcal{D} : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

Derivatives

$$\frac{\mathrm{d}ax^2}{\mathrm{d}x} \rightsquigarrow 2ax$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : \underbrace{f}_{\mathbb{R}\to\mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}\to\mathbb{R}}$$

$$\frac{\mathrm{d}}{\mathrm{d}x} : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

$$\mathcal{D} : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

$$\mathcal{D}\ \lambda x\ ax^2$$

Partial Derivatives

$$\frac{\partial a x^2 y^3}{\partial x} \qquad \frac{\partial a x^2 y^3}{\partial y}$$

Partial Derivatives

$$\frac{\partial a x^2 y^3}{\partial x} \qquad \frac{\partial a x^2 y^3}{\partial y}$$

$$\mathcal{D}\ \lambda x\ a x^2 y^3 \qquad \mathcal{D}\ \lambda y\ a x^2 y^3$$

Partial Derivatives

$$\frac{\partial a x^2 y^3}{\partial x} \qquad \frac{\partial a x^2 y^3}{\partial y}$$

$$\mathcal{D}\ \lambda x\ a x^2 y^3 \qquad \mathcal{D}\ \lambda y\ a x^2 y^3$$

$$\mathcal{D}_1\ \lambda(x, y)\ a x^2 y^3 \qquad \mathcal{D}_2\ \lambda(x, y)\ a x^2 y^3$$

Partial Derivatives

$$\frac{\partial a x^2 y^3}{\partial x} \qquad \frac{\partial a x^2 y^3}{\partial y}$$

$$\mathcal{D}\ \lambda x\ a x^2 y^3 \qquad \mathcal{D}\ \lambda y\ a x^2 y^3$$

$$\mathcal{D}_1\ \lambda(x,y)\ a x^2 y^3 \qquad \mathcal{D}_2\ \lambda(x,y)\ a x^2 y^3$$

$$\frac{\partial}{\partial x} : \underbrace{f}_{\mathbb{R}^n \rightarrow \mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}^n \rightarrow \mathbb{R}}$$

Partial Derivatives

$$\frac{\partial a x^2 y^3}{\partial x} \qquad \frac{\partial a x^2 y^3}{\partial y}$$

$$\mathcal{D}\ \lambda x\ a x^2 y^3 \qquad \mathcal{D}\ \lambda y\ a x^2 y^3$$

$$\mathcal{D}_1\ \lambda(x, y)\ a x^2 y^3 \qquad \mathcal{D}_2\ \lambda(x, y)\ a x^2 y^3$$

$$\frac{\partial}{\partial x} : \underbrace{f}_{\mathbb{R}^n \to \mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}^n \to \mathbb{R}}$$

$$\frac{\partial}{\partial x} : (\mathbb{R}^n \to \mathbb{R}) \to (\mathbb{R}^n \to \mathbb{R})$$

Partial Derivatives

$$\frac{\partial a x^2 y^3}{\partial x} \qquad \frac{\partial a x^2 y^3}{\partial y}$$

$$\mathcal{D}\ \lambda x\ a x^2 y^3 \qquad \mathcal{D}\ \lambda y\ a x^2 y^3$$

$$\mathcal{D}_1\ \lambda(x,y)\ a x^2 y^3 \qquad \mathcal{D}_2\ \lambda(x,y)\ a x^2 y^3$$

$$\frac{\partial}{\partial x} : \underbrace{f}_{\mathbb{R}^n \to \mathbb{R}} \mapsto \underbrace{f'}_{\mathbb{R}^n \to \mathbb{R}}$$

$$\frac{\partial}{\partial x} : (\mathbb{R}^n \to \mathbb{R}) \to (\mathbb{R}^n \to \mathbb{R})$$

$$\mathcal{D}_i : (\mathbb{R}^n \to \mathbb{R}) \to (\mathbb{R}^n \to \mathbb{R})$$

Gradients

$$\nabla\ f\ \mathbf{x} \quad = \quad (\mathcal{D}_1\ f\ \mathbf{x}), \ldots, (\mathcal{D}_n\ f\ \mathbf{x})$$

$$\nabla \quad : \quad (\mathbb{R}^n \rightarrow \mathbb{R}) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R}^n)$$

Jacobians

$$
\begin{aligned}
f &: \mathbb{R}^n \rightarrow \mathbb{R}^m \\
\mathbf{f} &: (\mathbb{R}^n \rightarrow \mathbb{R})^m \\
(\mathcal{J}\ f\ \mathbf{x})[i,j] &= (\nabla\ (\mathbf{f}[i]))[j] \\
\mathcal{J} &: (\mathbb{R}^n \rightarrow \mathbb{R}^m) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R}^{m\times n})
\end{aligned}
$$

The Chain Rule

$$(f \circ g)\ x = g\ (f\ x)$$

The Chain Rule

$$(f \circ g)\ x = g\ (f\ x)$$

$$\frac{\mathrm{d}g}{\mathrm{d}x} = \frac{\mathrm{d}g}{\mathrm{d}f}\frac{\mathrm{d}f}{\mathrm{d}x}$$

The Chain Rule

$$(f \circ g)\ x = g\ (f\ x)$$

$$\frac{\mathrm{d}g}{\mathrm{d}x} = \frac{\mathrm{d}g}{\mathrm{d}f}\frac{\mathrm{d}f}{\mathrm{d}x}$$

$$\mathcal{D}\ (f \circ g)\ x = (\mathcal{D}\ g\ (f\ x)) \times (\mathcal{D}\ f\ x)$$

The Chain Rule

$$(f \circ g)\ x = g\ (f\ x)$$

$$\frac{\mathrm{d}g}{\mathrm{d}x} = \frac{\mathrm{d}g}{\mathrm{d}f}\frac{\mathrm{d}f}{\mathrm{d}x}$$

$$\mathcal{D}\ (f \circ g)\ x = (\mathcal{D}\ g\ (f\ x)) \times (\mathcal{D}\ f\ x)$$

$$\mathcal{J}\ (f \circ g)\ \mathbf{x} = (\mathcal{J}\ g\ (f\ \mathbf{x})) \times (\mathcal{J}\ f\ \mathbf{x})$$

Matrix Transposition

$$\mathbf{A}^\top[i,j] \quad = \quad \mathbf{A}[j,i]$$

Matrix Transposition

$$\begin{aligned}\mathbf{A}^\top[i,j] &= \mathbf{A}[j,i]\\ (\mathbf{A}\times\mathbf{B})^\top &= \mathbf{B}^\top\times\mathbf{A}^\top\end{aligned}$$

Taylor Expansions

$$f(c+\varepsilon)=\frac{f(c)}{0!}+\frac{f'(c)}{1!}\varepsilon+\frac{f''(c)}{2!}\varepsilon^2+\cdots+\frac{f^{(i)}(c)}{i!}\varepsilon^i+\cdots$$

Taylor, B. (1715). *Methodus Incrementorum Directa et Inversa*, London.

The Essence of Forward-Mode AD

$$f(c+\varepsilon)=\frac{f(c)}{0!}+\frac{f'(c)}{1!}\varepsilon+\frac{f''(c)}{2!}\varepsilon^2+\cdots+\frac{f^{(i)}(c)}{i!}\varepsilon^i+\cdots$$

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,

The Essence of Forward-Mode AD

$$f(c+\varepsilon)=\frac{f(c)}{0!}+\frac{f'(c)}{1!}\varepsilon+\frac{f''(c)}{2!}\varepsilon^2+\cdots+\frac{f^{(i)}(c)}{i!}\varepsilon^i+\cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε,

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\; f\; c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε,

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1!

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1! (noop).

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1! (noop).

Key idea: Only need output to be a **finite** truncated power series $a + b\varepsilon$.

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1! (noop).

Key idea: Only need output to be a **finite** truncated power series $a + b\varepsilon$.

The input $c+\varepsilon$ is also a truncated power series.

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\; f\; c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1! (noop).

Key idea: Only need output to be a **finite** truncated power series $a + b\varepsilon$.

The input $c+\varepsilon$ is also a truncated power series.

Can do a *nonstandard interpretation* of f over truncated power series.

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1! (noop).

Key idea: Only need output to be a **finite** truncated power series $a+b\varepsilon$.

The input $c+\varepsilon$ is also a truncated power series.

Can do a *nonstandard interpretation* of f over truncated power series.

Preserves control flow: Augments original values with derivatives.

The Essence of Forward-Mode AD

$$f(c+\varepsilon) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}\varepsilon + \frac{f''(c)}{2!}\varepsilon^2 + \cdots + \frac{f^{(i)}(c)}{i!}\varepsilon^i + \cdots$$

To compute $\mathcal{D}\ f\ c$:

- evaluate f at the **term** $c+\varepsilon$ to get a **power series**,
- extract the coefficient of ε, and
- multiply by 1! (noop).

Key idea: Only need output to be a **finite** truncated power series $a + b\varepsilon$.

The input $c+\varepsilon$ is also a truncated power series.

Can do a *nonstandard interpretation* of f over truncated power series.

Preserves control flow: Augments original values with derivatives.

$(\mathcal{D}\ f)$ is $\mathcal{O}(1)$ relative to f (both space and time).

Arithmetic on Complex Numbers

$a + b\mathrm{i}$

Hamilton, W. R. (1837). Theory of conjugate functions, or algebraic couples; with a preliminary and elementary essay on algebra as the science of pure time, *Transactions of the Royal Irish Academy*, **17**(1):293–422.

Arithmetic on Complex Numbers

$a + b\mathrm{i}$
$\mathrm{i}^2 = -1$

Hamilton, W. R. (1837). Theory of conjugate functions, or algebraic couples; with a preliminary and elementary essay on algebra as the science of pure time, *Transactions of the Royal Irish Academy*, **17**(1):293–422.

Arithmetic on Complex Numbers

$a + b\mathrm{i}$
$\mathrm{i}^2 = -1$

$(a_1 + b_1\mathrm{i}) + (a_2 + b_2\mathrm{i}) = (a_1 + a_2) + (b_1 + b_2)\mathrm{i}$
$(a_1 + b_1\mathrm{i}) \times (a_2 + b_2\mathrm{i}) = (a_1 \times a_2 - b_1 \times b_2) + (a_1 \times b_2 + a_2 \times b_1)\mathrm{i}$

Hamilton, W. R. (1837). Theory of conjugate functions, or algebraic couples; with a preliminary and elementary essay on algebra as the science of pure time, *Transactions of the Royal Irish Academy*, **17**(1):293–422.

Arithmetic on Complex Numbers

$a + b\mathrm{i}$
$\mathrm{i}^2 = -1$

$(a_1 + b_1\mathrm{i}) + (a_2 + b_2\mathrm{i}) = (a_1 + a_2) + (b_1 + b_2)\mathrm{i}$
$(a_1 + b_1\mathrm{i}) \times (a_2 + b_2\mathrm{i}) = (a_1 \times a_2 - b_1 \times b_2) + (a_1 \times b_2 + a_2 \times b_1)\mathrm{i}$

$\langle a, b\rangle$

Hamilton, W. R. (1837). Theory of conjugate functions, or algebraic couples; with a preliminary and elementary essay on algebra as the science of pure time, *Transactions of the Royal Irish Academy*, **17**(1):293–422.

Arithmetic on Complex Numbers

$a + b\mathrm{i}$
$\mathrm{i}^2 = -1$

$$(a_1 + b_1\mathrm{i}) + (a_2 + b_2\mathrm{i}) = (a_1 + a_2) + (b_1 + b_2)\mathrm{i}$$
$$(a_1 + b_1\mathrm{i}) \times (a_2 + b_2\mathrm{i}) = (a_1 \times a_2 - b_1 \times b_2) + (a_1 \times b_2 + a_2 \times b_1)\mathrm{i}$$

$\langle a, b\rangle$

$$\langle a_1, b_1\rangle + \langle a_2, b_2\rangle = \langle (a_1 + a_2), (b_1 + b_2)\rangle$$
$$\langle a_1, b_1\rangle \times \langle a_2, b_2\rangle = \langle (a_1 \times a_2 - b_1 \times b_2), (a_1 \times b_2 + a_2 \times b_1)\rangle$$

Hamilton, W. R. (1837). Theory of conjugate functions, or algebraic couples; with a preliminary and elementary essay on algebra as the science of pure time, *Transactions of the Royal Irish Academy*, **17**(1):293–422.

Arithmetic on Dual Numbers

$x + x'\varepsilon$

Clifford, W. K. (1873). Preliminary Sketch of Bi-quaternions, *Proceedings of the London Mathematical Society*, **4**:381–95.

Arithmetic on Dual Numbers

$x + x'\varepsilon$
$\varepsilon^2 = 0$, but $\varepsilon \neq 0$

Clifford, W. K. (1873). Preliminary Sketch of Bi-quaternions, *Proceedings of the London Mathematical Society*, **4**:381–95.

Arithmetic on Dual Numbers

$x + x'\varepsilon$
$\varepsilon^2 = 0$, but $\varepsilon \neq 0$

$$(x_1 + x_1'\varepsilon) + (x_2 + x_2'\varepsilon) = (x_1 + x_2) + (x_1' + x_2')\varepsilon$$
$$(x_1 + x_1'\varepsilon) \times (x_2 + x_2'\varepsilon) = (x_1 \times x_2) + (x_1 \times x_2' + x_2 \times x_1')\varepsilon$$

Clifford, W. K. (1873). Preliminary Sketch of Bi-quaternions, *Proceedings of the London Mathematical Society*, **4**:381–95.

Arithmetic on Dual Numbers

$x + x'\varepsilon$
$\varepsilon^2 = 0$, but $\varepsilon \neq 0$

$$(x_1 + x'_1\varepsilon) + (x_2 + x'_2\varepsilon) = (x_1 + x_2) + (x'_1 + x'_2)\varepsilon$$
$$(x_1 + x'_1\varepsilon) \times (x_2 + x'_2\varepsilon) = (x_1 \times x_2) + (x_1 \times x'_2 + x_2 \times x'_1)\varepsilon$$

$\langle x, x' \rangle$

Clifford, W. K. (1873). Preliminary Sketch of Bi-quaternions, *Proceedings of the London Mathematical Society*, **4**:381–95.

Arithmetic on Dual Numbers

$x + x'\varepsilon$
$\varepsilon^2 = 0$, but $\varepsilon \neq 0$

$$(x_1 + x_1'\varepsilon) + (x_2 + x_2'\varepsilon) = (x_1 + x_2) + (x_1' + x_2')\varepsilon$$
$$(x_1 + x_1'\varepsilon) \times (x_2 + x_2'\varepsilon) = (x_1 \times x_2) + (x_1 \times x_2' + x_2 \times x_1')\varepsilon$$

$\langle x, x' \rangle$

$$\langle x_1, x_1' \rangle + \langle x_2, x_2' \rangle = \langle (x_1 + x_2), (x_1' + x_2') \rangle$$
$$\langle x_1, x_1' \rangle \times \langle x_2, x_2' \rangle = \langle (x_1 \times x_2), (x_1 \times x_2' + x_2 \times x_1') \rangle$$

Clifford, W. K. (1873). Preliminary Sketch of Bi-quaternions, *Proceedings of the London Mathematical Society*, **4**:381–95.

In differential geometry, dual numbers are known as (tangent) *bundles* of (primal) values x and their *tangents* $\overrightarrow{x'}$.

In differential geometry, dual numbers are known as (tangent) *bundles* of (primal) values x and their *tangents* $\overrightarrow{x}$.

$$x \triangleright \overrightarrow{x}$$

In differential geometry, dual numbers are known as (tangent) *bundles* of (primal) values x and their *tangents* $\overline{\acute{x}}$.

$$\overset{\rightharpoondown}{x} = x \triangleright \overline{\acute{x}}$$

Traditional Forward-Mode AD

$$\mathbb{R}^n \rightarrow \mathbb{R}^m \quad \rightsquigarrow \quad (\mathbb{R}^n \triangleright \overline{\mathbb{R}^{\cancel{n}}}) \rightarrow (\mathbb{R}^m \triangleright \overline{\mathbb{R}^{\cancel{m}}})$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Traditional Forward-Mode AD

$$
\begin{array}{lcl}
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R}^n \triangleright \overrightharpoon{\mathbb{R}^n}) \rightarrow (\mathbb{R}^m \triangleright \overrightharpoon{\mathbb{R}^m}) \\
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R} \triangleright \overrightharpoon{\mathbb{R}})^n \rightarrow (\mathbb{R} \triangleright \overrightharpoon{\mathbb{R}})^m
\end{array}
$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Nontraditional Forward-Mode AD

$$
\begin{array}{rcl}
\mathbb{R}^n \to \mathbb{R}^m & \rightsquigarrow & (\mathbb{R}^n \triangleright \overrightharpoon{\mathbb{R}^n}) \to (\mathbb{R}^m \triangleright \overrightharpoon{\mathbb{R}^m}) \\
\mathbb{R}^n \to \mathbb{R}^m & \rightsquigarrow & (\mathbb{R} \triangleright \overrightharpoon{\mathbb{R}})^n \to (\mathbb{R} \triangleright \overrightharpoon{\mathbb{R}})^m \\
\tau_1 \to \tau_2 & \rightsquigarrow & (\tau_1 \triangleright \overrightharpoon{\tau_1}) \to (\tau_2 \triangleright \overrightharpoon{\tau_2})
\end{array}
$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Nontraditional Forward-Mode AD

$$\begin{array}{rcl}
\mathbb{R}^n \longrightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R}^n \triangleright \overrightarrow{\mathbb{R}^n}) \longrightarrow (\mathbb{R}^m \triangleright \overrightarrow{\mathbb{R}^m}) \\
\mathbb{R}^n \longrightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R} \triangleright \overrightarrow{\mathbb{R}})^n \longrightarrow (\mathbb{R} \triangleright \overrightarrow{\mathbb{R}})^m \\
\tau_1 \longrightarrow \tau_2 & \rightsquigarrow & (\tau_1 \triangleright \overrightarrow{\tau_1}) \longrightarrow (\tau_2 \triangleright \overrightarrow{\tau_2}) \\
\tau_1 \longrightarrow \tau_2 & \rightsquigarrow & \overrightarrow{\tau_1} \longrightarrow \overrightarrow{\tau_2}
\end{array}$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Nontraditional Forward-Mode AD

$$
\begin{array}{rcl}
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R}^n \triangleright \overrightarrow{\mathbb{R}^n}) \rightarrow (\mathbb{R}^m \triangleright \overrightarrow{\mathbb{R}^m}) \\
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R} \triangleright \overrightarrow{\mathbb{R}})^n \rightarrow (\mathbb{R} \triangleright \overrightarrow{\mathbb{R}})^m \\
\tau_1 \rightarrow \tau_2 & \rightsquigarrow & (\tau_1 \triangleright \overrightarrow{\tau_1}) \rightarrow (\tau_2 \triangleright \overrightarrow{\tau_2}) \\
\overrightarrow{\mathcal{J}} : \tau_1 \rightarrow \tau_2 & \mapsto & \overrightarrow{\tau_1} \rightarrow \overrightarrow{\tau_2}
\end{array}
$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Nontraditional Forward-Mode AD

$$
\begin{array}{rcl}
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R}^n \rhd \overrightarrow{\mathbb{R}^n}) \rightarrow (\mathbb{R}^m \rhd \overrightarrow{\mathbb{R}^m}) \\
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R} \rhd \overrightarrow{\mathbb{R}})^n \rightarrow (\mathbb{R} \rhd \overrightarrow{\mathbb{R}})^m \\
\tau_1 \rightarrow \tau_2 & \rightsquigarrow & (\tau_1 \rhd \overrightarrow{\tau_1}) \rightarrow (\tau_2 \rhd \overrightarrow{\tau_2}) \\
\overrightarrow{\mathcal{J}} : \tau_1 \rightarrow \tau_2 & \mapsto & \overrightarrow{\tau_1} \rightarrow \overrightarrow{\tau_2} \\
\overrightarrow{\mathcal{J}} : \tau & \mapsto & \overrightarrow{\tau}
\end{array}
$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Nontraditional Forward-Mode AD

$$
\begin{array}{rcl}
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R}^n \rhd \overrightharpoon{\mathbb{R}^n}) \rightarrow (\mathbb{R}^m \rhd \overrightharpoon{\mathbb{R}^m}) \\
\mathbb{R}^n \rightarrow \mathbb{R}^m & \rightsquigarrow & (\mathbb{R} \rhd \overrightharpoon{\mathbb{R}})^n \rightarrow (\mathbb{R} \rhd \overrightharpoon{\mathbb{R}})^m \\
\tau_1 \rightarrow \tau_2 & \rightsquigarrow & (\tau_1 \rhd \overrightharpoon{\tau_1}) \rightarrow (\tau_2 \rhd \overrightharpoon{\tau_2}) \\
\overrightharpoon{\mathcal{J}} : \tau_1 \rightarrow \tau_2 & \mapsto & \overrightharpoon{\tau_1} \rightarrow \overrightharpoon{\tau_2} \\
\overrightharpoon{\mathcal{J}} : \tau & \mapsto & \overrightharpoon{\tau} \\
\mathcal{D}\, f\, x & = & \textbf{let}\ y_1 \rhd \overrightharpoon{y_2} = (\overrightharpoon{\mathcal{J}}\ f)\ x \rhd \overrightharpoon{1}\ \textbf{in}\ y_2
\end{array}
$$

Wengert, R. E. (1964). A simple automatic derivative evaluation program, *Communications of the ACM*, **7**(8):463–4.

Modularity

$$\nabla\, f\, \mathbf{x} \quad \triangleq \quad \frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}$$

Modularity

$$\nabla\; f\; \mathbf{x} \quad \triangleq \quad \frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}$$

$$\textsc{GradientDescent}\; f\; \mathbf{x}_0 \quad \triangleq \quad \ldots \mathbf{x}_{i+1} := \ldots \nabla\; f\; \mathbf{x}_i \ldots$$

Modularity

$$\nabla\ f\ \mathbf{x} \quad \triangleq \quad \frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \quad \triangleq \quad \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \ f \quad \triangleq \quad \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

Modularity

$$\nabla\ f\ \mathbf{x} \quad \triangleq \quad \frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \quad \triangleq \quad \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \quad \triangleq \quad \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \quad \triangleq \quad \boxed{\textit{classified}}$$

Modularity

$$\nabla\ f\ \mathbf{x} \triangleq \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right)$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

Modularity

$$\nabla\ f\ \mathbf{x} \triangleq \frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$r^* \triangleq \text{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ f\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \triangleright \overleftarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \triangleright \overleftarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$r^* \triangleq \text{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\mathrm{argmin}\ \ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\mathrm{critical}})^2$$

$$r^* \triangleq \mathrm{argmin}\ \ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$r^* \triangleq \text{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\operatorname{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$r^* \triangleq \operatorname{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\operatorname{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$r^* \triangleq \operatorname{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$r^* \triangleq \text{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla \overrightarrow{f}\; \mathbf{x} \triangleq (\overrightarrow{f}\; \mathbf{x} \rhd \overleftarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\; \mathbf{x} \rhd \overleftarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\; \overrightarrow{f}\; \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overrightarrow{f}\; \mathbf{x}_i \ldots$$

$$\mathrm{argmin}\; \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\; \overrightarrow{f}\; \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\; r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\; r \triangleq ((\textsc{NeutronFlux}\; r) - \textsc{NeutronFlux}_{\mathrm{critical}})^2$$

$$r^* \triangleq \mathrm{argmin}\; \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{Deviation}}$$

$$r^* \triangleq \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overrightarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ r \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ r \triangleq ((\textsc{NeutronFlux}\ r) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{Deviation}}$$

$$r^* \triangleq \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_1}), \ldots, (\overrightarrow{f}\ \mathbf{x} \rhd \overleftarrow{\mathbf{e}_n})$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overrightarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overrightarrow{f}\ \mathbf{x}_i \ldots$$

$$\mathrm{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \mathrm{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\rightsquigarrow} \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\operatorname{argmin}\ \overrightarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overrightarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\text{ADIFOR}}{\leadsto} \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\text{ADIFOR}}{\leadsto} \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \operatorname{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \quad \triangleq \quad \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \quad \triangleq \quad \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overrightarrow{f} \quad \triangleq \quad \ldots \textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \quad \triangleq \quad \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \quad \triangleq \quad ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \quad \triangleq \quad \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \quad \triangleq \quad \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \quad \triangleq \quad \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots$$

$$\text{argmin}\ \overleftarrow{f} \quad \triangleq \quad \ldots\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \quad \triangleq \quad \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \quad \triangleq \quad ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \quad \triangleq \quad \text{argmin}\ \overrightarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \quad \triangleq \quad \ldots\ \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \quad \triangleq \quad \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overleftarrow{f} \quad \triangleq \quad \ldots \textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \quad \triangleq \quad \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overrightarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \quad \triangleq \quad ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overrightarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \quad \triangleq \quad \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \quad \triangleq \quad \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \quad \triangleq \quad \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots$$

$$\text{argmin}\ \overleftarrow{f} \quad \triangleq \quad \ldots\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \quad \triangleq \quad \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \quad \triangleq \quad ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \quad \overset{\text{ADIFOR}}{\rightsquigarrow} \quad \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \quad \triangleq \quad \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overleftarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots \mathcal{H}\, f\ \mathbf{x}_i \ldots$$

$$\operatorname{argmin}\ \overleftarrow{f} \triangleq \ldots \textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \operatorname{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots \mathcal{H}\, f\ \mathbf{x}_i \ldots$$

$$\operatorname{argmin} \overleftarrow{f} \triangleq \ldots \textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \operatorname{argmin} \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$\nabla \overleftarrow{f}\ \mathbf{x}$	$\triangleq$	$\ldots \overleftarrow{f}\ \mathbf{x} \ldots$
$\mathcal{H}\ f\ \mathbf{x}$	$\triangleq$	
$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0$	$\triangleq$	$\ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots$
$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0$	$\triangleq$	$\ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots \mathcal{H}\ f\ \mathbf{x}_i \ldots$
$\mathrm{argmin}\ \overleftarrow{f}$	$\triangleq$	$\ldots \textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$
$\textsc{NeutronFlux}\ \mathbf{r}$	$\triangleq$	$\boxed{\textit{classified}}$
$\textsc{NeutronFlux}$	$\overset{\textsc{Tapenade}}{\rightsquigarrow}$	$\overleftarrow{\textsc{NeutronFlux}}$
$\textsc{Deviation}\ \mathbf{r}$	$\triangleq$	$((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\mathrm{critical}})^2$
$\textsc{Deviation}$	$\overset{\textsc{Tapenade}}{\rightsquigarrow}$	$\overleftarrow{\textsc{Deviation}}$
$\mathbf{r}^*$	$\triangleq$	$\mathrm{argmin}\ \overleftarrow{\textsc{Deviation}}$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\mathcal{H}\ f\ \mathbf{x} \triangleq \ldots \overrightarrow{\overleftarrow{f}}\ \ldots \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ \overleftarrow{f}\ \mathbf{x}_i \ldots \mathcal{H}\ f\ \mathbf{x}_i \ldots$$

$$\operatorname{argmin}\ \overleftarrow{f} \triangleq \ldots \textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \stackrel{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \stackrel{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \operatorname{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\mathcal{H}\ \overleftrightarrows{f}\ \mathbf{x} \triangleq \ldots \overleftrightarrows{f}\ \ldots\mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots\mathcal{H}\ f\ \mathbf{x}_i\ldots$$

$$\text{argmin}\ \overleftarrow{f} \triangleq \ldots\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\mathcal{H}\ \overleftrightarrows{f}\ \mathbf{x} \triangleq \ldots \overleftrightarrows{f}\ \ldots \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots \mathcal{H}\ \overleftrightarrows{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overleftarrow{f} \triangleq \ldots \textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \triangleq \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\mathcal{H}\ \overleftrightarrow{f}\ \mathbf{x} \triangleq \ldots\ \overleftrightarrow{f}\ \ldots\mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrow{f}\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots\mathcal{H}\ \overleftrightarrow{f}\ \mathbf{x}_i\ldots$$

$$\mathrm{argmin}\ \overleftarrow{f} \triangleq \ldots\textsc{NewtonsMethod}\ \overleftarrow{f}\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\leadsto} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\mathrm{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\leadsto} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \mathrm{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x} \ldots$$

$$\mathcal{H}\ \overleftrightarrows{f}\ \mathbf{x} \triangleq \ldots \overleftrightarrows{f}\ \ldots \mathbf{x} \ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrows{f}\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla \overleftarrow{f}\ \mathbf{x}_i \ldots \mathcal{H}\ \overleftrightarrows{f}\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ \overleftarrow{f} \triangleq \ldots \textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrows{f}\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\ \overleftarrow{f}\ \mathbf{x} \quad \triangleq \quad \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\mathcal{H}\ \overleftrightarrow{f}\ \mathbf{x} \quad \triangleq \quad \ldots\ \overleftrightarrow{f}\ \ldots\mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \quad \triangleq \quad \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrow{f}\ \mathbf{x}_0 \quad \triangleq \quad \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots\mathcal{H}\ \overleftrightarrow{f}\ \mathbf{x}_i\ldots$$

$$\text{argmin}\ \overleftarrow{f}\ \overleftrightarrow{f} \quad \triangleq \quad \ldots\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrow{f}\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \quad \triangleq \quad \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \quad \overset{\textsc{Tapenade}}{\leadsto} \quad \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \quad \triangleq \quad ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \quad \overset{\textsc{Tapenade}}{\leadsto} \quad \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \quad \triangleq \quad \text{argmin}\ \overleftarrow{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$\nabla\, \overleftarrow{f}\ \mathbf{x} \triangleq \ldots \overleftarrow{f}\ \mathbf{x}\ldots$$

$$\mathcal{H}\, \overset{\rightleftarrows}{f}\ \mathbf{x} \triangleq \ldots \overset{\rightleftarrows}{f}\ \ldots\mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\, \overleftarrow{f}\ \mathbf{x}_i\ldots$$

$$\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overset{\rightleftarrows}{f}\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\, \overleftarrow{f}\ \mathbf{x}_i\ldots\mathcal{H}\, \overset{\rightleftarrows}{f}\ \mathbf{x}_i\ldots$$

$$\text{argmin}\ \overleftarrow{f}\ \overset{\rightleftarrows}{f} \triangleq \ldots\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overset{\rightleftarrows}{f}\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{NeutronFlux} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{NeutronFlux}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\textsc{Deviation} \overset{\textsc{Tapenade}}{\rightsquigarrow} \overleftarrow{\textsc{Deviation}}$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \overleftarrow{\textsc{Deviation}}\ \overset{\rightleftarrows}{\textsc{Deviation}}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Breaking Modularity

$$
\begin{array}{lcl}
\nabla\ \overleftarrow{f}\ \mathbf{x} & \triangleq & \ldots\ \overleftarrow{f}\ \mathbf{x}\ldots \\
\mathcal{H}\ \overleftrightarrow{f}\ \mathbf{x} & \triangleq & \ldots\ \overleftrightarrow{f}\ \ldots\mathbf{x}\ldots \\
\textsc{GradientDescent}\ \overleftarrow{f}\ \mathbf{x}_0 & \triangleq & \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots \\
\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrow{f}\ \mathbf{x}_0 & \triangleq & \ldots\mathbf{x}_{i+1} := \ldots\nabla\ \overleftarrow{f}\ \mathbf{x}_i\ldots\mathcal{H}\ \overleftrightarrow{f}\ \mathbf{x}_i\ldots \\
\text{argmin}\ \overleftarrow{f}\ \overleftrightarrow{f} & \triangleq & \ldots\textsc{NewtonsMethod}\ \overleftarrow{f}\ \overleftrightarrow{f}\ \mathbf{x}_0\ldots \\
\textsc{NeutronFlux}\ \mathbf{r} & \triangleq & \boxed{\textit{classified}} \\
\textsc{NeutronFlux} & \overset{\textsc{Tapenade}}{\rightsquigarrow} & \overleftarrow{\textsc{NeutronFlux}} \\
\overleftarrow{\textsc{NeutronFlux}} & \overset{\textsc{Tapenade}}{\rightsquigarrow} & \overleftrightarrow{\textsc{NeutronFlux}} \\
\textsc{Deviation}\ \mathbf{r} & \triangleq & ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2 \\
\textsc{Deviation} & \overset{\textsc{Tapenade}}{\rightsquigarrow} & \overleftarrow{\textsc{Deviation}} \\
\overleftarrow{\textsc{Deviation}} & \overset{\textsc{Tapenade}}{\rightsquigarrow} & \overleftrightarrow{\textsc{Deviation}} \\
\mathbf{r}^* & \triangleq & \text{argmin}\ \overleftarrow{\textsc{Deviation}}\ \overleftrightarrow{\textsc{Deviation}}
\end{array}
$$

Fermi, E. (1946). *The Development of the first chain reaction pile.* Proceedings of the American Philosophy Society, **90**:20–4.

Restoring Modularity

$$\nabla\ f\ \mathbf{x} \triangleq$$

$$\mathcal{H}\ f\ \mathbf{x} \triangleq$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots \mathcal{H}\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Restoring Modularity

$$\nabla\ f\ \mathbf{x} \triangleq ((\overrightarrow{\mathcal{J}}\ f)\ \mathbf{x} \triangleright \overrightarrow{\mathbf{e}_1}), \ldots, ((\overrightarrow{\mathcal{J}}\ f)\ \mathbf{x} \triangleright \overrightarrow{\mathbf{e}_n})$$

$$\mathcal{H}\ f\ \mathbf{x} \triangleq$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots \mathcal{H}\ f\ \mathbf{x}_i \ldots$$

$$\mathrm{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\mathbf{r}^* \triangleq \mathrm{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Restoring Modularity

$$\nabla\ f\ \mathbf{x} \triangleq \ldots(\overleftarrow{\mathcal{J}}\ f)\ \mathbf{x}\ldots$$

$$\mathcal{H}\ f\ \mathbf{x} \triangleq$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots$$

$$\textsc{NewtonsMethod}\ f\ \mathbf{x}_0 \triangleq \ldots \mathbf{x}_{i+1} := \ldots \nabla\ f\ \mathbf{x}_i \ldots \mathcal{H}\ f\ \mathbf{x}_i \ldots$$

$$\text{argmin}\ f \triangleq \ldots \textsc{GradientDescent}\ f\ \mathbf{x}_0 \ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\mathbf{r}^* \triangleq \text{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Restoring Modularity

$$\nabla\, f\ \mathbf{x} \triangleq \ldots(\overleftarrow{\mathcal{J}}\ f)\ \mathbf{x}\ldots$$

$$\mathcal{H}\ f\ \mathbf{x} \triangleq \ldots(\overrightarrow{\mathcal{J}}\ (\overleftarrow{\mathcal{J}}\ f))\ldots\mathbf{x}\ldots$$

$$\textsc{GradientDescent}\ f\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\ f\ \mathbf{x}_i\ldots$$

$$\textsc{NewtonsMethod}\ f\ \mathbf{x}_0 \triangleq \ldots\mathbf{x}_{i+1} := \ldots\nabla\ f\ \mathbf{x}_i\ldots\mathcal{H}\ f\ \mathbf{x}_i\ldots$$

$$\operatorname{argmin}\ f \triangleq \ldots\textsc{NewtonsMethod}\ f\ \mathbf{x}_0\ldots$$

$$\textsc{NeutronFlux}\ \mathbf{r} \triangleq \boxed{\textit{classified}}$$

$$\textsc{Deviation}\ \mathbf{r} \triangleq ((\textsc{NeutronFlux}\ \mathbf{r}) - \textsc{NeutronFlux}_{\text{critical}})^2$$

$$\mathbf{r}^* \triangleq \operatorname{argmin}\ \textsc{Deviation}$$

Fermi, E. (1946). *The Development of the first chain reaction pile*. Proceedings of the American Philosophy Society, **90**:20–4.

Closure Properties

polynomials

Closure Properties

polynomials , product rule

Closure Properties

polynomials , product rule , quotient rule

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

chain rule

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Inductive case: *function composition*

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Inductive case: *function composition*

An inductive definition of the space of expressions

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Inductive case: *function composition*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) expression

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Inductive case: *function composition*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) expression

output space $\subseteq$ *input space*

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Inductive case: *function composition*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) expression

output space $\subseteq$ *input space*

Consequence 2: could take higher-order derivatives

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

chain rule

Inductive case: *lambda calculus*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) expression

output space $\subseteq$ *input space*

Consequence 2: could take higher-order derivatives

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

AD

Inductive case: *lambda calculus*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) expression

output space $\subseteq$ *input space*

Consequence 2: could take higher-order derivatives

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

AD

Inductive case: *lambda calculus*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) *program*

output space $\subseteq$ *input space*

Consequence 2: could take higher-order derivatives

Closure Properties

polynomials , product rule , quotient rule , transcendental functions

Base case: *arithmetic basis*

AD

Inductive case: *lambda calculus*

An inductive definition of the space of expressions

Consequence 1: could take the derivative of *any* (differentiable) *program*

output space $\subseteq$ *input space*

Consequence 2: could take higher-order derivatives *of programs*

FORTRAN

Backus, J. W. (1954). Preliminary Report: Specifications for the IBM Mathematical FORmula TRANslating System, FORTRAN, International Business Machines.

FORTRAN no recursion

Backus, J. W. (1954). Preliminary Report: Specifications for the IBM Mathematical FORmula TRANslating System, FORTRAN, International Business Machines.

A Brief History of Programming Languages

FORTRAN no recursion
ALGOL recursion

Naur, P. *et al.* (1963). Revised report on the algorithmic language Algol 60, *Communications of the ACM*, **6**(1):1–17.

A Brief History of Programming Languages

FORTRAN no recursion

ALGOL recursion

no arrays of arrays

Naur, P. *et al.* (1963). Revised report on the algorithmic language Algol 60, *Communications of the ACM*, **6**(1):1–17.

A Brief History of Programming Languages

FORTRAN no recursion

ALGOL recursion
no arrays of arrays

ALGOL-68 recursion
arrays of arrays

van Wijngaarden, A. *et al.* 1976, Revised Report on the Algorithmic Language Algol 68, Springer-Verlag.

A Brief History of Programming Languages

FORTRAN no recursion

ALGOL recursion
no arrays of arrays

ALGOL-68 recursion
arrays of arrays
no escaping closures

van Wijngaarden, A. *et al.* 1976, Revised Report on the Algorithmic Language Algol 68, Springer-Verlag.

A Brief History of Programming Languages

FORTRAN no recursion

ALGOL recursion
no arrays of arrays

ALGOL-68 recursion
arrays of arrays
no escaping closures

SCHEME recursion
arrays of arrays
escaping closures

Sussman, G. J. and Steele, Jr., G. L. (1975). *Scheme: an Interpreter for Extended Lambda Calculus*, MIT AI memo 349.

Game Theory

needs work

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ.

Shooting Method

needs work

Goldstine, A. (1946). *Report on the ENIAC (Electronic Numerical Integrator and Computer)*, Moore School of Electrical Engineering, University of Pennsylvania.

Cathode Ray Tubes

needs work

Sprague, C. S. and George, R. H. (1939). Cathod Ray Deflecting Electrode. US Patent 2,161,437.

George, R. H. (1940). Cathod Ray Tube. US Patent 2,222,942.

STALIN∇ VS. SCHEME

	Language/Implementation										
Example	STALIN∇	IKARUS	STALIN	SCHEME->C	CHICKEN	BIGLOO	GAMBIT	LARCENY	MZC	MZSCHEME	SCMUTILS
`saddle`	1.00	61.71	94.85	112.90	233.35	175.07	130.50	184.90	613.45	720.41	705.10
`particle`	1.00	146.96	248.00	308.34	609.30	501.59	351.20	537.07	1453.19	1868.88	1512.90

	Language/Implementation				
Example	STALIN∇	MLTON	SML/NJ	OCAML	GHC
`saddle`	1.00	11.19	16.68	21.25	31.08
`particle`	1.00	33.13	40.34	58.53	74.56

The State of the Art Regarding Closure Properties

The State of the Art Regarding Closure Properties

ADIFOR

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph

The State of the Art Regarding Closure Properties

ADIFOR	miscomputes call graph confuses nesting with recursion

The State of the Art Regarding Closure Properties

ADIFOR	miscomputes call graph confuses nesting with recursion cannot handle indirect function calls

The State of the Art Regarding Closure Properties

ADIFOR	miscomputes call graph confuses nesting with recursion cannot handle indirect function calls generates incorrect derivative code

The State of the Art Regarding Closure Properties

ADIFOR	miscomputes call graph confuses nesting with recursion cannot handle indirect function calls generates incorrect derivative code gives wrong answer without warning

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph
confuses nesting with recursion
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer without warning
to get it to work
- specialize indirect function calls
- copy code
- split code into separate files
- manually edit both input code and generated code

TAPENADE

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph
confuses nesting with recursion
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer without warning
to get it to work
- specialize indirect function calls
- copy code
- split code into separate files
- manually edit both input code and generated code

TAPENADE generates code with syntax errors

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph
confuses nesting with recursion
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer without warning
to get it to work
- specialize indirect function calls
- copy code
- split code into separate files
- manually edit both input code and generated code

TAPENADE generates code with syntax errors
cannot handle indirect function calls

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE
- generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE
- generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer (with aliasing warning)

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE
- generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer (with aliasing warning)
- to get it to work
  - specialize indirect function calls
  - manually edit both input code and generated code

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE
- generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer (with aliasing warning)
- to get it to work
  - specialize indirect function calls
  - manually edit both input code and generated code

ADIC

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE
- generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer (with aliasing warning)
- to get it to work
  - specialize indirect function calls
  - manually edit both input code and generated code

ADIC
- generates code with syntax errors

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer (with aliasing warning)
- to get it to work
  - specialize indirect function calls
  - manually edit both input code and generated code

ADIC generates code with syntax errors
- cannot transform its own output

The State of the Art Regarding Closure Properties

ADIFOR
- miscomputes call graph
- confuses nesting with recursion
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer without warning
- to get it to work
  - specialize indirect function calls
  - copy code
  - split code into separate files
  - manually edit both input code and generated code

TAPENADE
- generates code with syntax errors
- cannot handle indirect function calls
- generates incorrect derivative code
- gives wrong answer (with aliasing warning)
- to get it to work
  - specialize indirect function calls
  - manually edit both input code and generated code

ADIC
- generates code with syntax errors
- cannot transform its own output

FADBAD++

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph
confuses nesting with recursion
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer without warning
to get it to work
- specialize indirect function calls
- copy code
- split code into separate files
- manually edit both input code and generated code

TAPENADE generates code with syntax errors
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer (with aliasing warning)
to get it to work
- specialize indirect function calls
- manually edit both input code and generated code

ADIC generates code with syntax errors
cannot transform its own output

FADBAD++ cannot take derivatives of arbitrary unmodified C++ programs

The State of the Art Regarding Closure Properties

ADIFOR miscomputes call graph
confuses nesting with recursion
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer without warning
to get it to work
- specialize indirect function calls
- copy code
- split code into separate files
- manually edit both input code and generated code

TAPENADE generates code with syntax errors
cannot handle indirect function calls
generates incorrect derivative code
gives wrong answer (with aliasing warning)
to get it to work
- specialize indirect function calls
- manually edit both input code and generated code

ADIC generates code with syntax errors
cannot transform its own output

FADBAD++ cannot take derivatives of arbitrary unmodified C++ programs
to get it to work
- rewrite code using templates

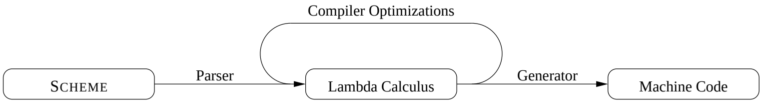
STALIN∇ vs C++ and FORTRAN

	Language/Implementation			
Example	STALIN∇	FADBAD++	ADIFOR	TAPENADE
`saddle`	1.00	5.71	0.49	0.73
`particle`	1.00	30.07	0.85	1.76

Static Floating-Point Instruction Density

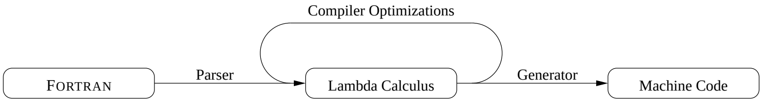
	Language/Implementation			
Example	STALIN∇	FADBAD++	ADIFOR	TAPENADE
`saddle`	16.9%	1.3%	9.3%	7.8%
`particle`	20.9%	1.6%	7.0%	4.4%

Lambda the Ultimate Intermediate Language



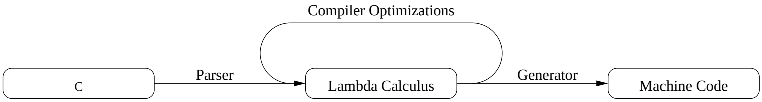
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language



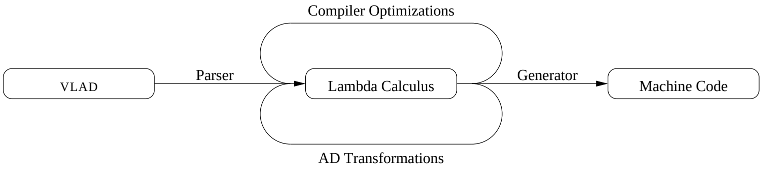
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language



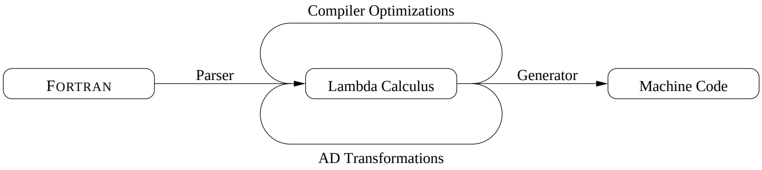
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language *for AD*



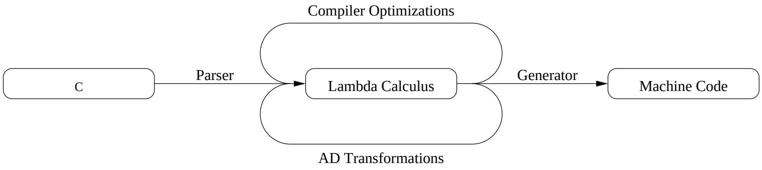
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language *for AD*



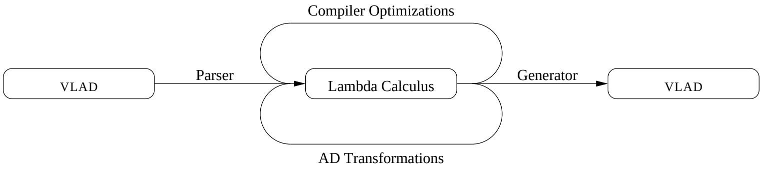
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language *for AD*



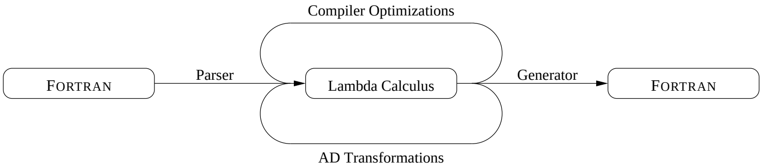
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language *for AD*



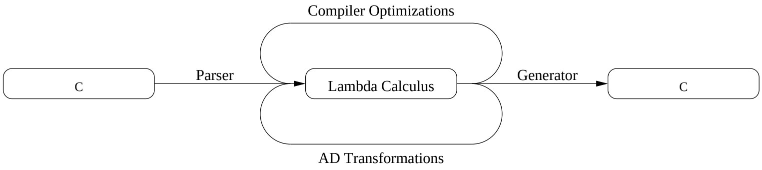
Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language *for AD*



Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

Lambda the Ultimate Intermediate Language *for AD*



Steele, Jr., G. L. and Sussman, G. J. (1976). *Lambda, the Ultimate Imperative*, MIT AI memo 353.

⋮

|

Marvin Lee Minsky

|

Gerald Jay Sussman

|

Guy Lewis Steele, Jr.

Something for Matthias Blume

$$\texttt{(zero}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(zero}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$

Something for Matthias Blume

$$\texttt{(zero}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(zero}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(primal}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$

Something for Matthias Blume

$$\texttt{(zero }\langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(zero }v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(primal }\langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal }v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(tangent }\langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(tangent }v_1\texttt{)},\ldots\},\ldots\rangle$$

Something for Matthias Blume

$$\texttt{(zero}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(zero}\ v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(primal}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal}\ v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(tangent}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(tangent}\ v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(j*}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j*}\ v_1\texttt{)},\ldots\},\ldots\rangle$$

Something for Matthias Blume

$$\texttt{(zero}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(zero}\ v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(primal}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal}\ v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(tangent}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(tangent}\ v_1\texttt{)},\ldots\},\ldots\rangle$$
$$\texttt{(j*}\ \langle\{x_1 \mapsto v_1,\ldots\},\ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j*}\ v_1\texttt{)},\ldots\},\ldots\rangle$$

ditto for our *entire* AD basis (`perturb`, `unperturb`, `bundle`, `sensitize`, `unsensitize`, `plus`, `*j`, and `*j-inverse`)

Something for Matthias Blume

$$\texttt{(zero}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(zero}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(primal}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(tangent}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(tangent}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j*}\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$

ditto for our *entire* AD basis (`perturb`, `unperturb`, `bundle`, `sensitize`, `unsensitize`, `plus`, `*j`, and `*j-inverse`)

$$\texttt{(map-closure}\ f\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(}f\ v_1\texttt{)}, \ldots\}, \ldots\rangle$$

Siskind, J. M. and Pearlmutter, B. A. (2007). First-Class Nonstandard Interpretations by Opening Closures, *Proceedings of the 34th Symposium on Principles of Programming Languages*, 71–6.

Something for Robby Findler

$$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \ldots\rangle$$

Something for Robby Findler

$$\texttt{(j* }\langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j* }v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(j* }\langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j* }v_1\texttt{)}, \ldots\}, \overrightarrow{e}\rangle$$

Something for Robby Findler

$$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \ldots\rangle$$
$$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \overrightharpoon{e}\rangle$$
$$(\texttt{*j}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{*j}\ v_1), \ldots\}, \overleftharpoon{e}\rangle$$
$$(\texttt{primal}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overrightharpoon{e}\rangle) = \langle\{x_1 \mapsto (\texttt{primal}\ v_1), \ldots\}, e\rangle$$
$$(\texttt{*j-inverse}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overleftharpoon{e}\rangle) =$$
$$\langle\{x_1 \mapsto (\texttt{*j-inverse}\ v_1), \ldots\}, e\rangle$$

Something for Robby Findler

$$\texttt{(j* } \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j* } v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(j* } \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j* } v_1\texttt{)}, \ldots\}, \overrightarrow{e}\rangle$$
$$\texttt{(*j } \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(*j } v_1\texttt{)}, \ldots\}, \overleftarrow{e}\rangle$$
$$\texttt{(primal } \langle\{x_1 \mapsto v_1, \ldots\}, \overrightarrow{e}\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal } v_1\texttt{)}, \ldots\}, e\rangle$$
$$\texttt{(*j-inverse } \langle\{x_1 \mapsto v_1, \ldots\}, \overleftarrow{e}\rangle\texttt{)} =$$
$$\langle\{x_1 \mapsto \texttt{(*j-inverse } v_1\texttt{)}, \ldots\}, e\rangle$$

$$\texttt{(map-closure-and-transform } f\ \mathcal{T}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} =$$
$$\langle\{x_1 \mapsto \texttt{(}f\ v_1\texttt{)}, \ldots\}, \mathcal{T}(e)\rangle$$

Something for Robby Findler

$$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \ldots\rangle$$
$$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \overrightarrow{e}\rangle$$
$$(\texttt{*j}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{*j}\ v_1), \ldots\}, \overleftarrow{e}\rangle$$
$$(\texttt{primal}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overrightarrow{e}\rangle) = \langle\{x_1 \mapsto (\texttt{primal}\ v_1), \ldots\}, e\rangle$$
$$(\texttt{*j-inverse}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overleftarrow{e}\rangle) =$$
$$\langle\{x_1 \mapsto (\texttt{*j-inverse}\ v_1), \ldots\}, e\rangle$$

$(\texttt{map-closure-and-transform}\ f\ \mathcal{T}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) =$
$\langle\{x_1 \mapsto (f\ \ v_1), \ldots\}, \mathcal{T}(e)\rangle$ But what is $\mathcal{T}$?

Something for Robby Findler

$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \ldots\rangle$
$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \overrightarrow{e}\rangle$
$(\texttt{*j}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{*j}\ v_1), \ldots\}, \overleftarrow{e}\rangle$
$(\texttt{primal}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overrightarrow{e}\rangle) = \langle\{x_1 \mapsto (\texttt{primal}\ v_1), \ldots\}, e\rangle$
$(\texttt{*j-inverse}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overleftarrow{e}\rangle) =$
$\langle\{x_1 \mapsto (\texttt{*j-inverse}\ v_1), \ldots\}, e\rangle$

$(\texttt{map-closure-and-transform}\ f\ \mathcal{T}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) =$
$\langle\{x_1 \mapsto (f\ \ v_1), \ldots\}, \mathcal{T}(e)\rangle$ But what is $\mathcal{T}$?

- a procedural macro, `defmacro`

Something for Robby Findler

$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \ldots\rangle$
$(\texttt{j*}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{j*}\ v_1), \ldots\}, \overrightarrow{e}\rangle$
$(\texttt{*j}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) = \langle\{x_1 \mapsto (\texttt{*j}\ v_1), \ldots\}, \overleftarrow{e}\rangle$
$(\texttt{primal}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overrightarrow{e}\rangle) = \langle\{x_1 \mapsto (\texttt{primal}\ v_1), \ldots\}, e\rangle$
$(\texttt{*j-inverse}\ \langle\{x_1 \mapsto v_1, \ldots\}, \overleftarrow{e}\rangle) =$
$\langle\{x_1 \mapsto (\texttt{*j-inverse}\ v_1), \ldots\}, e\rangle$

$(\texttt{map-closure-and-transform}\ f\ \mathcal{T}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle) =$
$\langle\{x_1 \mapsto (f\ \ v_1), \ldots\}, \mathcal{T}(e)\rangle$ But what is $\mathcal{T}$?

- a procedural macro, `defmacro`
- a hygienic macro, `syntax-rules`, `syntax-case`, ...

Something for Robby Findler

$$\texttt{(j* }\langle\{x_1 \mapsto v_1, \ldots\}, \ldots\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j* } v_1\texttt{)}, \ldots\}, \ldots\rangle$$
$$\texttt{(j* }\langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(j* } v_1\texttt{)}, \ldots\}, \overrightarrow{e}\rangle$$
$$\texttt{(*j }\langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(*j } v_1\texttt{)}, \ldots\}, \overleftarrow{e}\rangle$$
$$\texttt{(primal }\langle\{x_1 \mapsto v_1, \ldots\}, \overrightarrow{e}\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(primal } v_1\texttt{)}, \ldots\}, e\rangle$$
$$\texttt{(*j-inverse }\langle\{x_1 \mapsto v_1, \ldots\}, \overleftarrow{e}\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(*j-inverse } v_1\texttt{)}, \ldots\}, e\rangle$$

$$\texttt{(map-closure-and-transform } f\ \mathcal{T}\ \langle\{x_1 \mapsto v_1, \ldots\}, e\rangle\texttt{)} = \langle\{x_1 \mapsto \texttt{(}f\ \ v_1\texttt{)}, \ldots\}, \mathcal{T}(e)\rangle$$ But what is $\mathcal{T}$?

- a procedural macro, `defmacro`
- a hygienic macro, `syntax-rules`, `syntax-case`, …
- a rewrite system, PLT REDEX

Something for Dave MacQueen and John Reppy

zero $: \tau \rightarrow \tau$
perturb $: \tau \rightarrow \acute{\overline{\tau}}$
unperturb $: \acute{\overline{\tau}} \rightarrow \tau$
primal $: \overrightarrow{\tau} \rightarrow \tau$
tangent $: \overrightarrow{\tau} \rightarrow \acute{\overline{\tau}}$
bundle $: \tau \times \acute{\overline{\tau}} \rightarrow \overrightarrow{\tau}$
j* $: \tau \rightarrow \overrightarrow{\tau}$
sensitize $: \tau \rightarrow \grave{\overline{\tau}}$
unsensitize $: \grave{\overline{\tau}} \rightarrow \tau$
plus $: \tau \times \tau \rightarrow \tau$
*j $: \tau \rightarrow \overleftarrow{\tau}$
*j-inverse $: \overleftarrow{\tau} \rightarrow \tau$

$$\frac{f : \tau_1 \rightarrow \tau_2 \qquad\qquad x : \tau_1}{(fx) : \tau_2}$$

(j* f)(j* x)
(j* $f : \tau$) $: \overrightarrow{\tau}$
(j* $f : \tau_1 \rightarrow \tau_2$) $: \overrightarrow{\tau_1 \rightarrow \tau_2}$

Something for Jean Utke

TAPENADE 2.1 User's Guide (p. 72):

> ***10. KNOWN PROBLEMS AND DEVELOPMENTS TO COME***
> ***10.4 Pointers and dynamic allocation***
> *For example, how should we handle a memory deallocation in the reverse mode? During the reverse sweep, the memory must be reallocated somehow, and the pointers must point back into this reallocated memory. Finding the more efficient way to handle this is still an open problem.*

`http://www-unix.mcs.anl.gov/~utke/OpenAD/`:

> ***4. Language-coverage and library handling in adjoint code***
> ***2. language concepts (e.g., array arithmetic, pointers and dynamic memory allocation, polymorphism):***
> *Many language concepts, in particular those found in object-oriented languages, have never been considered in the context of automatic adjoint code generation. We are aware of several hard theoretical and technical problems that need to be*

Something else for Jean Utke

Review 3
significance: 2/5 originality: 2/5
The authors present the "inability to nest" as a "central limitation" that prevents current AD tools from being truly automatic. [...] What constitutes a "central limitation" is, however, a rather subjective criterion. There are other problems that are in my view much more critical to the practical use of AD tools. Take, for instance, adjoining parallelized models.
Dear Dr. Pearlmutter,
[...] The problems with nesting transformations of the current Fortran/C - AD tools - contrary claims on their respective websites not withstanding - are a known fact but there nesting has not been a priority. [...] The suggested road map connecting vlad and stalingrad to other language front-ends is in our view not necessary. [...] It merely highlights the fact that after criticizing the other tools vlad and stalingrad cannot readily be used either to differentiate Fortran or C programs, do reverse mode with checkpointing, cross