

Automatic Matrix Differentiation

Anju Kambadur¹ and Peder A. Olsen²

¹Bloomberg L.P., ²IBM TJ Watson Research Center,

A Brief Introduction

- Work in the AI and NLP group at Bloomberg.
 - Current Focus:
 - Sequence learning, spectral methods, and sparse models.
 - Numerical linear algebra/scientific computing.
 - Open-source initiatives.
 - Machine translation.
 - Knowledge representation (ontologies).
 - Information extraction.
- Was at IBM Research till earlier this year.

Covariance: A Motivating Example

$$\text{Gaussian pdf} = (2\pi)^{-\frac{n}{2}} \det(\mathbf{\Sigma})^{-\frac{1}{2}} e^{(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Covariance: A Motivating Example

$$\text{Gaussian pdf} = (2\pi)^{-\frac{n}{2}} \det(\mathbf{\Sigma})^{-\frac{1}{2}} e^{(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Tikhonov regularized maximum log-likelihood for learning the covariance $\mathbf{\Sigma}$ given an empirical covariance \mathbf{S} is:

$$f(\mathbf{\Sigma}) = -\log \det(\mathbf{\Sigma}) - \text{trace}(\mathbf{S}\mathbf{\Sigma}^{-1}) - \|\mathbf{\Sigma}^{-1}\|_F^2.$$

Covariance: A Motivating Example

$$\text{Gaussian pdf} = (2\pi)^{-\frac{n}{2}} \det(\Sigma)^{-\frac{1}{2}} e^{(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)}$$

Tikhonov regularized maximum log-likelihood for learning the covariance Σ given an empirical covariance S is:

$$f(\Sigma) = -\log \det(\Sigma) - \text{trace}(S\Sigma^{-1}) - \|\Sigma^{-1}\|_F^2.$$

- To optimize $f(\Sigma)$, using first-order methods we need $f'(\Sigma)$.
- To use second-order methods, we also need $f''(\Sigma)$.

What are $f'(\Sigma)$ and $f''(\Sigma)$?

Objective

Theoretical Aspect

Differentiation of matrix functions is still not well understood.

- No simple “calculus” for computing matrix derivatives.
- Matrix derivatives of simple functions can be complicated.

Build on known results to define a matrix-derivative calculus.

Objective

Software Aspect

Matrix differentiation made simple for humans and computers

- Symbolic differentiation should be made simple.
- Optimizing matrix valued functions should be efficient:
 - Both computation and storage.

We are developing Automatic Matrix Differentiation (AMD), an open-source C++ library to accomplish this. The code can be downloaded from <https://github.com/pkambadu/AMD>

AMD in Action

$$f(\Sigma) = -\log \det(\Sigma) - \text{trace}(\mathbf{S}\Sigma^{-1}) - \|\Sigma^{-1}\|_F^2.$$

```
$ ./SymbolicCalculator 'logdet(X)'
```

Function: $\log(\det(X))$

Derivative: $\text{inv}(X)'$

```
$ ./SymbolicCalculator 'trace(S*inv(X))'
```

Function: $\text{trace}(\mathbf{S}\text{inv}(X))$

Derivative: $(-(\text{inv}(X) * (\mathbf{S} * \text{inv}(X))))'$

```
$ ./SymbolicCalculator 'trace(inv(trans(X)*X))'
```

Function: $\text{trace}(\text{inv}(X' * X))$

Derivative: $((X * (-(\text{inv}(X' * X) * \text{inv}(X' * X)))) + ((-(\text{inv}(X' * X) * \text{inv}(X' * X))) * X'))'$

Benefits of AMD

In A Nutshell

AMD can be used to optimize scalar-matrix functions:

- Function is $f(\mathbf{X}) : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}$
- Gradient is $f'(\mathbf{X}) : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}^{d \times d}$
- Hessian is $f''(\mathbf{X}) : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}^{d^2 \times d^2}$

Additionally, AMD can be used for:

- Optimizing proximal gradient methods.
- Taylor expansions for sensitivity analysis.
- Lots more

Overview

- Motivation
- Introduction
 - Numerical, symbolic, and automatic differentiation
 - Matrix differentiation
 - `trace()` and `logdet()`
 - Forward and reverse mode differentiation
 - Rules of Matrix Differentiation
- Theoretical contributions
 - `Boxproduct()` (addition to `vec()` and `Kronecker()`)
 - Matrix derivative identities
- Software contributions
 - AMD (<https://github.com/pkambadu/AMD>)

Terminology

Classical calculus:

- **scalar-scalar functions** ($f : \mathbb{R} \rightarrow \mathbb{R}$)
- **scalar-vector functions** ($f : \mathbb{R}^d \rightarrow \mathbb{R}$).

Matrix calculus:

- **scalar-matrix functions** ($f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$)
- **matrix-matrix functions** ($f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{k \times l}$).

Note:

- 1 Scalar-matrix derivatives require matrix-matrix derivatives
- 2 Matrix-matrix derivatives should be computed implicitly

How can we compute the derivative?

Numerical Use finite difference formulae for differentiation (e.g., $f'(x) \approx \frac{f(x+h)-f(x)}{h}$). These methods are simple to program, but lose half of all significant digits.

Symbolic Give the symbolic representation of the derivative without saying how best to implement the derivative; these are as accurate as they come.

Algorithmic Something between numeric and symbolic differentiation. The derivative implementation is computed from the code for $f(x)$.

Numerical Differentiation

From First-Principles

Estimate the gradient by looking at the rate of change.

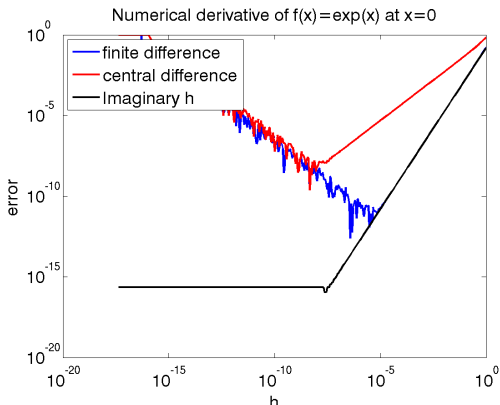
- Finite Difference: $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
- Central Difference: $f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$
- Imaginary Difference: $f'(x) \approx \text{Im} \left(\frac{f(x+ih)}{h} \right)$

For $h < \epsilon x$, where ϵ is the machine precision, these approximations become zero. Thus h has to be chosen with care.

Numerical Differentiation

Imaginary Differentiation Error

The graph shows the error in approximating the derivative for $f(x) = e^x$ as a function of h .



Symbolic Differentiation

Introduction

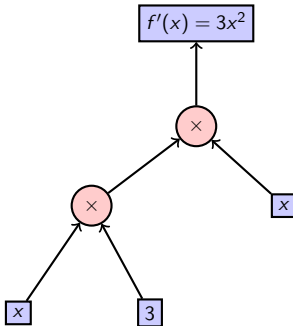
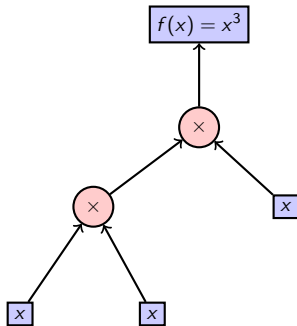
Calculate derivatives by symbolic manipulation of expressions.

| Function | Derivative |
|-----------------------------|---|
| x^p | px^{p-1} |
| $\sin \theta$ | $\cos \theta$ |
| $\cos \theta$ | $-\sin \theta$ |
| $\log x$ | x^{-1} |
| e^x | e^x |
| $\frac{d(af(x)+bg(x))}{dx}$ | $a \frac{d(f(x))}{dx} + b \frac{d(g(x))}{dx}$ |
| $\frac{d(f(x)g(x))}{dx}$ | $f(x) \frac{d(g(x))}{dx} + g(x) \frac{d(f(x))}{dx}$ |
| $\frac{d(f(g(x)))}{dx}$ | $\frac{d(f(x))}{dg} \frac{d(g(x))}{dx}$ |

- Minimal set of well-known functions and their derivatives.
- Rules for computing derivatives of compositions of functions.

Symbolic Scalar/Vector Differentiation

$$f(x) = x^3; \quad f'(x) = 3x^2;$$



Symbolic Differentiation

Drawbacks

- Extremely useful for simple functions.
 - $f(x) = x^3$.
- Not helpful for complex functions: may be several pages.
 - $f(\Sigma) = -\log \det(\Sigma) - \text{trace}(\mathbf{S}\Sigma^{-1}) - \|\Sigma^{-1}\|_F^2$
- Does not tell us how to compute the derivative.
 - Especially because we can reuse computations.

Automatic Differentiation

Algorithmic/Automatic Differentiation (AD) uses the software representation of a function to obtain an efficient method for calculating its derivatives. These derivatives can be of arbitrary order and are analytic in nature (do not have any truncation error).

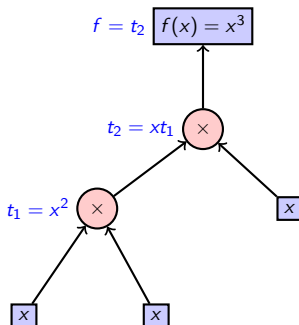
B. Bell – author of *CppAD*

- No truncation error — like Symbolic, unlike Numerical.
- Gives an algorithm for computing the derivative.
 - Often, (conveniently) in terms of the constituent functions.

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

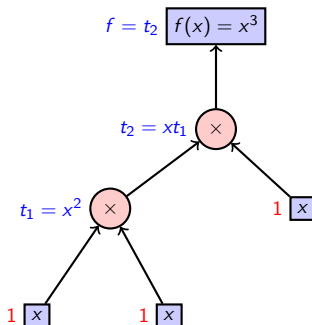
Compute $f'(x)$; $f(x) = x^3$



Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

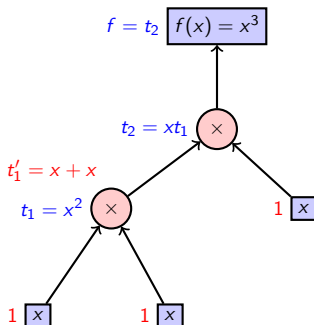


Forward Mode: Inside-out Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

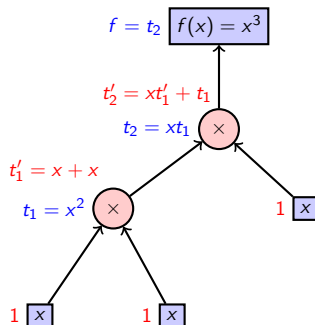


Forward Mode: Inside-out Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

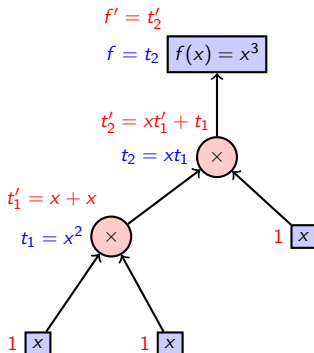


Forward Mode: Inside-out Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

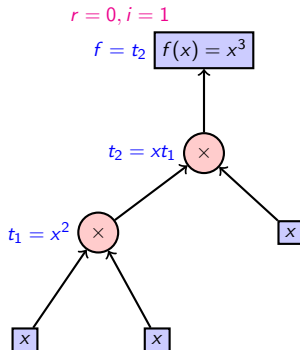


Forward Mode: Inside-out Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

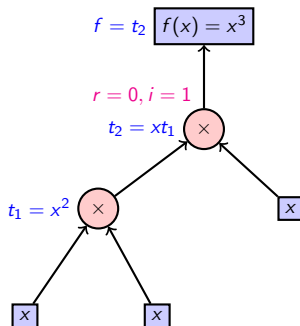


Reverse Mode: Outside-in Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

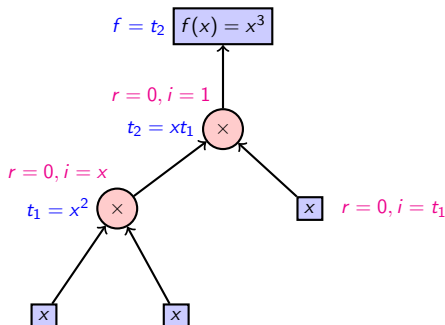


Reverse Mode: Outside-in Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

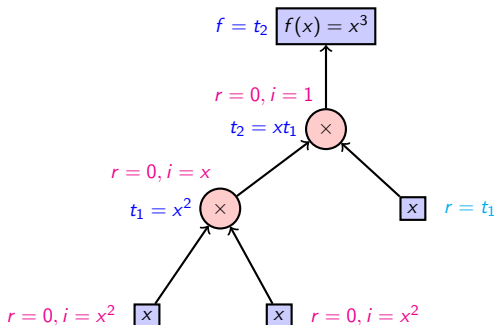


Reverse Mode: Outside-in Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$

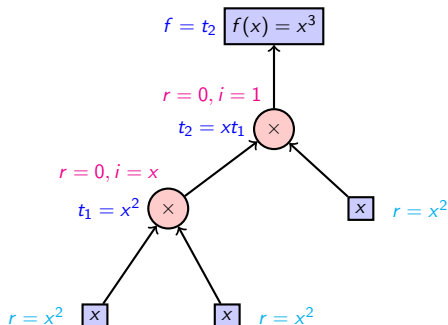


Reverse Mode: Outside-in Application

Automatic Scalar/Vector Differentiation

Forward and Reverse Mode

Compute $f'(x)$; $f(x) = x^3$



Reverse Mode: Outside-in Application

Automatic Scalar/Vector Differentiation

Notes on Reverse Mode Differentiation

Reverse Mode Differentiation (1971):

- Chain rule applied in reverse order of function evaluation.
- Same technique as:
 - Back-propagation algorithm (1969) for neural networks.
 - Forward-backward algorithm for training HMMs (1970).
- Applies to very general classes of functions.

Automatic Scalar/Vector Differentiation

Computational Cost of Differentiation

Under quite realistic assumptions the evaluation of a gradient requires never more than five times the effort of evaluating the underlying function by itself.

Andreas Griewank (1988)

Scalar-Matrix Derivatives

Definition

Derivative is the rate of change of $f(\mathbf{X})$ w.r.t each x_{ij}

For $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ the scalar-matrix derivative is defined to be:

$$\frac{\partial f}{\partial \mathbf{X}} \stackrel{\text{def}}{=} \begin{pmatrix} \frac{\partial f}{\partial x_{11}} & \frac{\partial f}{\partial x_{12}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \frac{\partial f}{\partial x_{21}} & \frac{\partial f}{\partial x_{22}} & \cdots & \frac{\partial f}{\partial x_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_{m1}} & \frac{\partial f}{\partial x_{m2}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{pmatrix}.$$

Scalar-Matrix Derivatives

Derivative of trace \mathbf{X}

Derivative is the rate of change of $f(\mathbf{X})$ w.r.t each x_{ij}

$$\begin{aligned} f(\mathbf{X}) &= \text{trace} \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \\ f'(\mathbf{X}) &= \begin{pmatrix} \frac{\partial(x_{11}+x_{22})}{\partial x_{11}} & \frac{\partial(x_{11}+x_{22})}{\partial x_{12}} \\ \frac{\partial(x_{11}+x_{22})}{\partial x_{21}} & \frac{\partial(x_{11}+x_{22})}{\partial x_{22}} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \mathbf{I} \end{aligned}$$

Scalar-Matrix Derivatives

Derivative of $\log \det \mathbf{X}$

Derivative is the rate of change of $f(\mathbf{X})$ w.r.t each x_{ij}

$$\begin{aligned}
 f(\mathbf{X}) &= \log \det \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \\
 f'(\mathbf{X}) &= \begin{pmatrix} \frac{\partial(\log x_{11}x_{22} - x_{12}x_{21})}{\partial x_{11}} & \frac{\partial(\log x_{11}x_{22} - x_{12}x_{21})}{\partial x_{12}} \\ \frac{\partial(\log x_{11}x_{22} - x_{12}x_{21})}{\partial x_{21}} & \frac{\partial(\log x_{11}x_{22} - x_{12}x_{21})}{\partial x_{22}} \end{pmatrix} \\
 &= \frac{1}{x_{11}x_{22} - x_{12}x_{21}} \begin{pmatrix} \frac{\partial(x_{11}x_{22} - x_{12}x_{21})}{\partial x_{11}} & \frac{\partial(x_{11}x_{22} - x_{12}x_{21})}{\partial x_{12}} \\ \frac{\partial(x_{11}x_{22} - x_{12}x_{21})}{\partial x_{21}} & \frac{\partial(x_{11}x_{22} - x_{12}x_{21})}{\partial x_{22}} \end{pmatrix} \\
 &= \frac{1}{\det \mathbf{X}} \begin{pmatrix} x_{22} & -x_{21} \\ -x_{12} & x_{11} \end{pmatrix} \\
 &= \mathbf{X}^{-\top}
 \end{aligned}$$

Matrix-Matrix Derivatives

Why

Derivative is the rate of change of $\mathbf{F}(\mathbf{X})$ w.r.t each x_{ij}

Scalar-matrix derivative of $f(\mathbf{G}(\mathbf{X}))$ requires the information in the matrix-matrix derivative $\frac{\partial \mathbf{G}}{\partial \mathbf{X}}$. For example:

- $\text{trace} \left(\left((\mathbf{I} + \mathbf{X})^{-1} \right) \mathbf{X} \right)$

Desiderata: The derivative of a matrix-matrix function should be a matrix, so that a convenient chain-rule can be established.

Matrix-Matrix Derivative

The Arrangement Problem

- For $\mathbf{F}(\mathbf{X}) = \mathbf{Y}; \mathbf{X} \in \mathbb{R}^{m \times n}, \mathbf{Y} \in \mathbb{R}^{k \times l}, \mathbf{F}'(\mathbf{X}) \in \mathbb{R}^{mn \times kl}$
 - $\mathbf{F}'(\mathbf{X})$ is a 4-D structure.
 - How do you represent $\mathbf{F}'(\mathbf{X})$ in 2-D?

$$\mathbf{F}(\mathbf{X}) = \mathbf{X}$$

$$\mathbf{F}' \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix}$$

$\mathbf{F}'(\mathbf{X})$ does not have nice structure

Matrix-Matrix Derivatives

The vec Operator

For $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{vec}(\mathbf{A})$ is the column-stacked vector of \mathbf{A}

$$\text{vec}(\mathbf{A}) = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \\ a_{12} \\ \vdots \\ a_{mn} \end{pmatrix}$$

The Matrix-Matrix Derivative

Definition

We define the matrix-matrix derivative to be:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \stackrel{\text{def}}{=} \frac{\partial \text{vec}(\mathbf{F}^\top)}{\partial \text{vec}^\top(\mathbf{X}^\top)} = \begin{pmatrix} \frac{\partial f_{11}}{\partial x_{11}} & \frac{\partial f_{11}}{\partial x_{12}} & \cdots & \frac{\partial f_{11}}{\partial x_{mn}} \\ \frac{\partial f_{12}}{\partial x_{11}} & \frac{\partial f_{12}}{\partial x_{12}} & \cdots & \frac{\partial f_{12}}{\partial x_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{kl}}{\partial x_{11}} & \frac{\partial f_{kl}}{\partial x_{12}} & \cdots & \frac{\partial f_{kl}}{\partial x_{mn}} \end{pmatrix}.$$

- This rule seems arcane at first.
 - Results derivatives being well-behaved matrices.

The Matrix-Matrix Derivative

Definition

We define the matrix–matrix derivative to be:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \stackrel{\text{def}}{=} \frac{\partial \text{vec}(\mathbf{F}^\top)}{\partial \text{vec}^\top(\mathbf{X}^\top)} = \begin{pmatrix} \frac{\partial f_{11}}{\partial x_{11}} & \frac{\partial f_{11}}{\partial x_{12}} & \cdots & \frac{\partial f_{11}}{\partial x_{mn}} \\ \frac{\partial f_{12}}{\partial x_{11}} & \frac{\partial f_{12}}{\partial x_{12}} & \cdots & \frac{\partial f_{12}}{\partial x_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{kl}}{\partial x_{11}} & \frac{\partial f_{kl}}{\partial x_{12}} & \cdots & \frac{\partial f_{kl}}{\partial x_{mn}} \end{pmatrix}.$$

- This rule seems arcane at first.
 - Results derivatives being well-behaved matrices.

The Matrix-Matrix Derivative

Definition

We define the matrix–matrix derivative to be:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \stackrel{\text{def}}{=} \frac{\partial \text{vec}(\mathbf{F}^\top)}{\partial \text{vec}^\top(\mathbf{X}^\top)} = \begin{pmatrix} \frac{\partial f_{11}}{\partial x_{11}} & \frac{\partial f_{11}}{\partial x_{12}} & \cdots & \frac{\partial f_{11}}{\partial x_{mn}} \\ \frac{\partial f_{12}}{\partial x_{11}} & \frac{\partial f_{12}}{\partial x_{12}} & \cdots & \frac{\partial f_{12}}{\partial x_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{kl}}{\partial x_{11}} & \frac{\partial f_{kl}}{\partial x_{12}} & \cdots & \frac{\partial f_{kl}}{\partial x_{mn}} \end{pmatrix}.$$

- This rule seems arcane at first.
 - Results derivatives being well-behaved matrices.

Matrix-Matrix Derivative

The Arrangement Problem ($\mathbf{F}(\mathbf{X}) = \mathbf{X}$)

$$\begin{aligned}
 \mathbf{F}' \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} &= \frac{\partial \text{vec}(\mathbf{F}^\top)}{\partial \text{vec}^\top(\mathbf{X}^\top)} = \frac{\begin{pmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{pmatrix}}{\begin{pmatrix} x_{11} & x_{12} & x_{21} & x_{22} \end{pmatrix}} \\
 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 &= \mathbf{I}_{d^2 \times d^2} \\
 &= \mathbf{I}_{d \times d} \otimes \mathbf{I}_{d \times d}
 \end{aligned}$$

Direct Matrix Products

Derivatives of *Linear* Matrix-Matrix Functions

- ① A matrix–matrix derivative is a matrix outer-product:

$$F(\mathbf{X}) = \text{trace}(\mathbf{A}\mathbf{X})\mathbf{B}, \quad \frac{\partial F}{\partial \mathbf{X}} = \text{vec}(\mathbf{B}^\top) \text{vec}^\top(\mathbf{A}).$$

- ② A matrix–matrix derivative is a Kronecker product:

$$F(\mathbf{X}) = \mathbf{A}\mathbf{X}\mathbf{B}, \quad \frac{\partial F}{\partial \mathbf{X}} = \mathbf{A} \otimes \mathbf{B}^\top.$$

- ③ A matrix–matrix derivative **box product**.

$$F(\mathbf{X}) = \mathbf{A}\mathbf{X}^\top \mathbf{B}, \quad \frac{\partial F}{\partial \mathbf{X}} = \mathbf{A} \boxtimes \mathbf{B}^\top.$$

Matrix-Matrix Derivatives

Direct Matrix Products

Direct Matrix Product, $\mathbf{X} = \mathbf{A} \circledast \mathbf{B}$, is a matrix such that:

$$x_{(i_1 i_2)(i_3 i_4)} = a_{i_{\sigma(1)} i_{\sigma(2)}} b_{i_{\sigma(3)} i_{\sigma(4)}}.$$

- $(i_1 i_2)$ is shorthand for $i_1 n_2 + i_2 - 1$ where $i_2 \in \{1, 2, \dots, n_2\}$.
- σ is a permutation over \mathbb{Z}_4

There are 24 possible permutations that can occur

Matrix-Matrix Derivatives

Direct Matrix Product

- 8 can be expressed using matrix outer products.

$$\begin{array}{cccc} \text{vec}(\mathbf{A})\text{vec}(\mathbf{B})^\top & \text{vec}(\mathbf{A}^\top)\text{vec}(\mathbf{B})^\top & \text{vec}(\mathbf{B})\text{vec}(\mathbf{A})^\top & \text{vec}(\mathbf{B}^\top)\text{vec}(\mathbf{A})^\top \\ \text{vec}(\mathbf{A})\text{vec}(\mathbf{B}^\top)^\top & \text{vec}(\mathbf{A}^\top)\text{vec}(\mathbf{B}^\top)^\top & \text{vec}(\mathbf{B})\text{vec}(\mathbf{A}^\top)^\top & \text{vec}(\mathbf{B}^\top)\text{vec}(\mathbf{A}^\top)^\top \end{array}$$

- 8 can be expressed using Kronecker products.

$$\begin{array}{cccc} \mathbf{A} \otimes \mathbf{B} & \mathbf{A}^\top \otimes \mathbf{B} & \mathbf{A} \otimes \mathbf{B}^\top & \mathbf{A}^\top \otimes \mathbf{B}^\top \\ \mathbf{B} \otimes \mathbf{A} & \mathbf{B}^\top \otimes \mathbf{A} & \mathbf{B} \otimes \mathbf{A}^\top & \mathbf{B}^\top \otimes \mathbf{A}^\top \end{array}$$

- Remaining 8 can be expressed using Box products.

$$\begin{array}{cccc} \mathbf{A} \boxtimes \mathbf{B} & \mathbf{A}^\top \boxtimes \mathbf{B} & \mathbf{A} \boxtimes \mathbf{B}^\top & \mathbf{A}^\top \boxtimes \mathbf{B}^\top \\ \mathbf{B} \boxtimes \mathbf{A} & \mathbf{B}^\top \boxtimes \mathbf{A} & \mathbf{B} \boxtimes \mathbf{A}^\top & \mathbf{B}^\top \boxtimes \mathbf{A}^\top \end{array}$$

Theoretical Contributions

Quick Recap

So far, we have seen:

- Automatic differentiation of scalars/vectors.
- Forward- and reverse-mode differentiation.
- Definition of derivatives for:
 - scalar-matrix and matrix-matrix functions.
- Direct matrix products
 - Kronecker and matrix-outer products (known).
 - **Box products (new contribution).**

Theoretical Contributions

Designing AMD

- Design box product.
 - Needed to compute all 24 direct matrix products.
- Establish derivatives for the functions we care about:
 - Scalar-matrix: [trace](#) and [log-determinant](#).
 - Matrix-matrix: [+](#), [-](#), [*](#), [.*](#), [transpose](#) and [inverse](#).
- Establish chain rule for function compositions.
- Figure out the mode:
 - Forward-mode (intuitive but expensive).
 - matrix-matrix (counter-intuitive but efficient).
- Do everything without blowing up memory.

Box Product

Definition

Let $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$

Definition (Kronecker Product)

$\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{(m_1 m_2) \times (n_1 n_2)}$ is defined by $(\mathbf{A} \otimes \mathbf{B})_{(i-1)m_2+j, (k-1)n_2+l} = a_{ik} b_{jl} = (\mathbf{A} \otimes \mathbf{B})_{(ij)(kl)}$.

Definition (Box Product)

$\mathbf{A} \boxtimes \mathbf{B} \in \mathbb{R}^{(m_1 m_2) \times (n_1 n_2)}$ is defined by $(\mathbf{A} \boxtimes \mathbf{B})_{(i-1)m_2+j, (k-1)n_1+l} = a_{i/l} b_{jk} = (\mathbf{A} \boxtimes \mathbf{B})_{(ij)(kl)}$.

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22}
 \end{array} \right)
 \end{array}
 \qquad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22}
 \end{array} \right)
 \end{array}
 \quad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \begin{pmatrix}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22}
 \end{pmatrix}
 \end{array}
 \qquad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \begin{pmatrix}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{pmatrix}
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22}
 \end{array} \right)
 \end{array}
 \quad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22}
 \end{array} \right)
 \end{array}
 \qquad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22}
 \end{array} \right)
 \end{array}
 \qquad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 \textcolor{green}{a}_{11}b_{21} & a_{11}b_{22} & \textcolor{green}{a}_{12}b_{21} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 \textcolor{green}{a}_{21}b_{21} & a_{21}b_{22} & \textcolor{green}{a}_{22}b_{21} & a_{22}b_{22}
 \end{array} \right)
 \end{array}
 \quad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 \textcolor{red}{a}_{11}b_{21} & \textcolor{red}{a}_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 \textcolor{red}{a}_{21}b_{21} & \textcolor{red}{a}_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

An example Kronecker and box product

Consider two 2×2 matrices, **A** and **B**:

$$\begin{array}{c}
 \mathbf{A} \otimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\
 a_{11}b_{21} & \textcolor{green}{a}_{11}b_{22} & a_{12}b_{21} & \textcolor{green}{a}_{12}b_{22} \\
 a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\
 a_{21}b_{21} & \textcolor{green}{a}_{21}b_{22} & a_{22}b_{21} & \textcolor{green}{a}_{22}b_{22}
 \end{array} \right)
 \end{array}
 \qquad
 \begin{array}{c}
 \mathbf{A} \boxtimes \mathbf{B} \\
 \left(\begin{array}{cccc}
 a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\
 a_{11}b_{21} & a_{12}b_{21} & \textcolor{red}{a}_{11}b_{22} & \textcolor{red}{a}_{12}b_{22} \\
 a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\
 a_{21}b_{21} & a_{22}b_{21} & \textcolor{red}{a}_{21}b_{22} & \textcolor{red}{a}_{22}b_{22}
 \end{array} \right)
 \end{array}$$

Box Product

Some Identities

The box product behaves similarly to the Kronecker product:

1 Vector Multiplication:

$$(\mathbf{B}^\top \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{AXB}) \quad (\mathbf{B}^\top \boxtimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{AX}^\top \mathbf{B})$$

2 Matrix Multiplication:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}) \quad (\mathbf{A} \boxtimes \mathbf{B})(\mathbf{C} \boxtimes \mathbf{D}) = (\mathbf{AD}) \otimes (\mathbf{BC})$$

3 Inverse and Transpose:

$$\begin{aligned} (\mathbf{A} \otimes \mathbf{B})^{-1} &= \mathbf{A}^{-1} \otimes \mathbf{B}^{-1} & (\mathbf{A} \boxtimes \mathbf{B})^{-1} &= \mathbf{B}^{-1} \boxtimes \mathbf{A}^{-1} \\ (\mathbf{A} \otimes \mathbf{B})^\top &= \mathbf{A}^\top \otimes \mathbf{B}^\top & (\mathbf{A} \boxtimes \mathbf{B})^\top &= \mathbf{B}^\top \boxtimes \mathbf{A}^\top \end{aligned}$$

4 Mixed Products:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \boxtimes \mathbf{D}) = (\mathbf{AC}) \boxtimes (\mathbf{BD}) \quad (\mathbf{A} \boxtimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AD}) \otimes (\mathbf{BC})$$

Box Product

Some Identities

Let $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$ then

- **Trace:**

$$\text{trace}(\mathbf{A} \otimes \mathbf{B}) = \text{trace}(\mathbf{A}) \text{trace}(\mathbf{B}) \quad \text{trace}(\mathbf{A} \boxtimes \mathbf{B}) = \text{trace}(\mathbf{AB})$$

- **Determinant:** Here $m_1 = n_1$ and $m_2 = n_2$ is required

$$\det(\mathbf{A} \otimes \mathbf{B}) = (\det(\mathbf{A}))^{m_2} (\det(\mathbf{B}))^{m_1}$$

$$\det(\mathbf{A} \boxtimes \mathbf{B}) = (-1)^{\binom{m_1}{2} \binom{m_2}{2}} (\det(\mathbf{A}))^{m_2} (\det(\mathbf{B}))^{m_1}$$

- **Associativity:**

$$(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) \quad (\mathbf{A} \boxtimes \mathbf{B}) \boxtimes \mathbf{C} = \mathbf{A} \boxtimes (\mathbf{B} \boxtimes \mathbf{C}),$$

but not for mixed products. In general we have

$$(\mathbf{A} \otimes \mathbf{B}) \boxtimes \mathbf{C} \neq \mathbf{A} \otimes (\mathbf{B} \boxtimes \mathbf{C}) \quad (\mathbf{A} \boxtimes \mathbf{B}) \otimes \mathbf{C} \neq \mathbf{A} \boxtimes (\mathbf{B} \otimes \mathbf{C}).$$

Box Product

Identity Box Products

$\mathbf{I}_m \boxtimes \mathbf{I}_n$ is permutation matrix; Let $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$:

Orthonormal: $(\mathbf{I}_m \boxtimes \mathbf{I}_n)^\top (\mathbf{I}_m \boxtimes \mathbf{I}_n) = \mathbf{I}_{mn}$

Transposition: $(\mathbf{I}_{m_1} \boxtimes \mathbf{I}_{n_1}) \text{vec}(\mathbf{A}) = \text{vec}(\mathbf{A}^\top)$

Connector: Converting a Kronecker product to a box product:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{I}_{n_1} \boxtimes \mathbf{I}_{n_2}) = \mathbf{A} \boxtimes \mathbf{B}.$$

Converting a box product to a Kronecker product:

$$(\mathbf{A} \boxtimes \mathbf{B})(\mathbf{I}_{n_2} \boxtimes \mathbf{I}_{n_1}) = \mathbf{A} \otimes \mathbf{B}.$$

Box Product

A Wolf in Sheep's Clothing

- Notation for identity box products is new, but:
 - Physics: $\mathbf{I}_m \boxtimes \mathbf{I}_n \rightarrow \mathbf{T}_{m,n}$
 - Computer Science: $\mathbf{I}_m \boxtimes \mathbf{I}_n \rightarrow$ stride permutation
 - Statistics: $\mathbf{I}_m \boxtimes \mathbf{I}_n \rightarrow$ perfect shuffle
- Box-product can express complex identities compactly.

Scalar-Matrix Derivatives

First some simple trace identities:

$$f(\mathbf{X}) = \text{trace}(\mathbf{X})$$

$$f'(\mathbf{X}) = \mathbf{I}$$

$$f(\mathbf{X}) = \text{trace}(\mathbf{A}\mathbf{X})$$

$$f'(\mathbf{X}) = \mathbf{A}^\top$$

$$f(\mathbf{X}) = \text{trace}(\mathbf{A}\mathbf{X}^\top)$$

$$f'(\mathbf{X}) = \mathbf{A}$$

Now, some simple log det identities:

$$f(\mathbf{X}) = \log \det(\mathbf{X})$$

$$f'(\mathbf{X}) = \mathbf{X}^{-\top}$$

$$f(\mathbf{X}) = \log \det(\mathbf{A}\mathbf{X})$$

$$f'(\mathbf{X}) = \mathbf{X}^{-\top}$$

Matrix-Matrix Derivative

Basic Differentiation Identities

Let $\mathbf{X} \in \mathbb{R}^{m \times n}$.

- **Identity:**

$$\mathbf{F}(\mathbf{X}) = \mathbf{X}, \mathbf{F}'(\mathbf{X}) = \mathbf{I}_{mn}$$

- **Transpose:**

$$\mathbf{F}(\mathbf{X}) = \mathbf{X}^\top, \mathbf{F}'(\mathbf{X}) = \mathbf{I}_m \boxtimes \mathbf{I}_n$$

- **Chain Rule:**

$$\mathbf{F}(\mathbf{X}) = \mathbf{G}(\mathbf{H}(\mathbf{X})), \quad \mathbf{F}'(\mathbf{X}) = \mathbf{G}'(\mathbf{H}(\mathbf{X}))\mathbf{H}'(\mathbf{X})$$

- **Product Rule:**

$$\mathbf{F}(\mathbf{X}) = \mathbf{G}(\mathbf{X})\mathbf{H}(\mathbf{X}), \quad \mathbf{F}'(\mathbf{X}) = (\mathbf{I} \otimes \mathbf{H}^\top(\mathbf{X}))\mathbf{G}'(\mathbf{X}) + (\mathbf{G}(\mathbf{X}) \otimes \mathbf{I})\mathbf{H}'(\mathbf{X})$$

Derivative Identities

Assume $\mathbf{X} \in \mathbb{R}^{m \times m}$ is a square matrix.

- **Square:** $\mathbf{F}(\mathbf{X}) = \mathbf{X}^2$, $\mathbf{F}'(\mathbf{X}) = \mathbf{I}_m \otimes \mathbf{X}^\top + \mathbf{X} \otimes \mathbf{I}_m$.
- **Inverse:** $\mathbf{F}(\mathbf{X}) = \mathbf{X}^{-1}$, $\mathbf{F}'(\mathbf{X}) = -\mathbf{X}^{-1} \otimes \mathbf{X}^{-\top}$.
- **+Transpose:** $\mathbf{F}(\mathbf{X}) = \mathbf{X}^{-\top}$, $\mathbf{F}'(\mathbf{X}) = -\mathbf{X}^{-\top} \boxtimes \mathbf{X}^{-1}$.
- **Square Root:** $\mathbf{F}(\mathbf{X}) = \mathbf{X}^{1/2}$, $\mathbf{F}'(\mathbf{X}) = \left(\mathbf{I} \otimes (\mathbf{X}^{1/2})^\top + \mathbf{X}^{1/2} \otimes \mathbf{I} \right)^{-1}$.
- **Integer Power:** $\mathbf{F}(\mathbf{X}) = \mathbf{X}^k$, $\mathbf{F}'(\mathbf{X}) = \sum_{i=0}^{k-1} \mathbf{X}^i \otimes (\mathbf{X}^{k-1-i})^\top$.

Matrix-Matrix Derivative

Setting Up Chain Rule($\mathbf{F}(\mathbf{X}) = \text{trace}(\mathbf{X})$)

$$\mathbf{F}(\mathbf{X}) = \text{trace}(\mathbf{X})$$

$$\begin{aligned}\mathbf{F}'(\mathbf{X}) &= \frac{\partial}{\partial \mathbf{X}} (\text{trace}(\mathbf{X})) \frac{\partial \mathbf{X}}{\partial \mathbf{X}} \\ &= \mathbf{I} (\mathbf{I} \otimes \mathbf{I}) \\ &= \mathbb{R}^{m \times m} (\mathbb{R}^{m^2 \times m^2})\end{aligned}$$

- This does not make any sense.
- We know that:
 - For scalar-matrix functions: $\mathbf{F}'(\mathbf{X}) \in \mathbb{R}^{m \times n}$
 - For matrix-matrix functions: $\mathbf{F}'(\mathbf{X}) \in \mathbb{R}^{m^2 \times n^2}$

Matrix-Matrix Derivative

Setting Up Chain Rule($\mathbf{F}(\mathbf{X}) = \text{trace}(\mathbf{X})$)

$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$

$$\begin{aligned} \mathbf{F}(\mathbf{X}) &= \text{trace}(\mathbf{X}) \\ \text{vec} \left((\mathbf{F}'(\mathbf{X}))^\top \right) &= \left(\left(\frac{\partial \mathbf{X}}{\partial \mathbf{X}} \right)^\top \right) \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{X}} (\text{trace}(\mathbf{X})) \right)^\top \right) \\ &= (\mathbf{I} \otimes \mathbf{I})^\top \text{vec} \left(\mathbf{I}^\top \right) \\ &= (\mathbf{I} \otimes \mathbf{I}) \text{vec} (\mathbf{I}) \\ &= \text{vec} (\mathbf{I}) \\ \mathbf{F}'(\mathbf{X}) &= \mathbf{I} \in \mathbb{R}^{m \times m} \end{aligned}$$

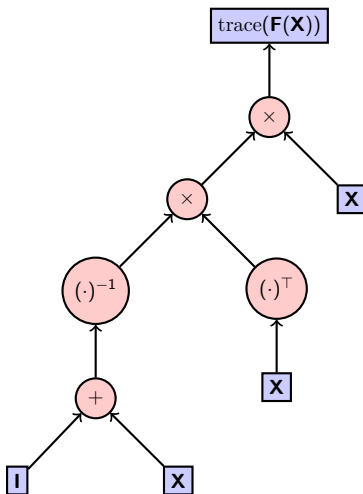
Matrix-Matrix Derivative

Setting Up Chain Rule ($\mathbf{F}(\mathbf{X}) = \log \det(\mathbf{X})$)

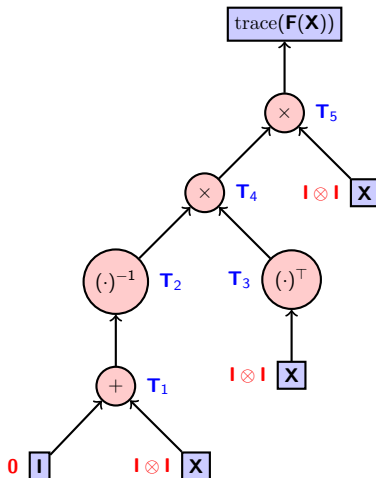
$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$

$$\begin{aligned} \mathbf{F}(\mathbf{X}) &= \log \det(\mathbf{X}) \\ \text{vec} \left((\mathbf{F}'(\mathbf{X}))^\top \right) &= \left(\left(\frac{\partial \mathbf{X}}{\partial \mathbf{X}} \right)^\top \right) \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{X}} (\log \det(\mathbf{X})) \right)^\top \right) \\ &= (\mathbf{I} \otimes \mathbf{I})^\top \text{vec} \left((\mathbf{X}^{-\top})^\top \right) \\ &= (\mathbf{I} \otimes \mathbf{I}) \text{vec} \left(\mathbf{X}^{-1} \right) = \text{vec} \left(\mathbf{X}^{-1} \right) \\ \mathbf{F}'(\mathbf{X}) &= \mathbf{X}^{-\top} \in \mathbb{R}^{m \times m} \end{aligned}$$

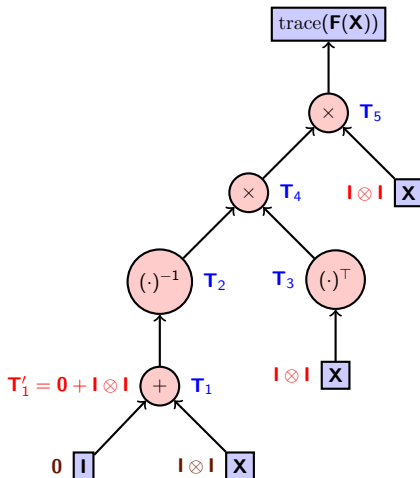
Compute $f(\mathbf{X}) = \text{trace}(\mathbf{F}(\mathbf{X})) = \text{trace}(((\mathbf{I} + \mathbf{X})^{-1}\mathbf{X}^\top)\mathbf{X})$



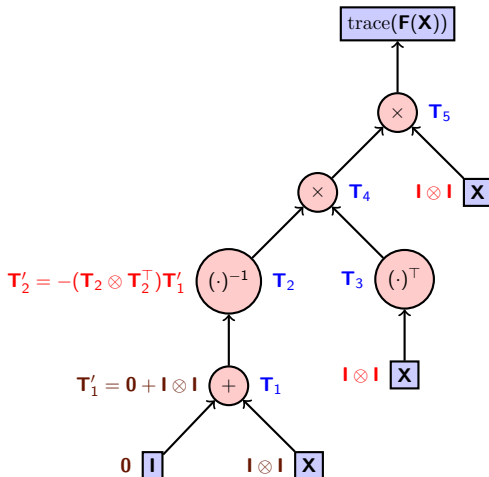
Forward mode computation of $f'(\mathbf{X})$:



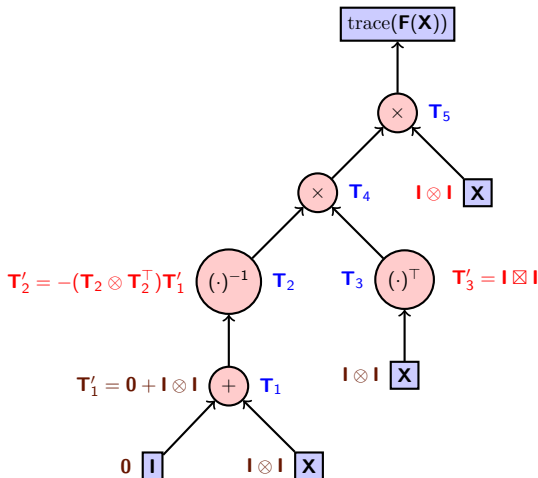
$$(G + H)' = G' + H'$$



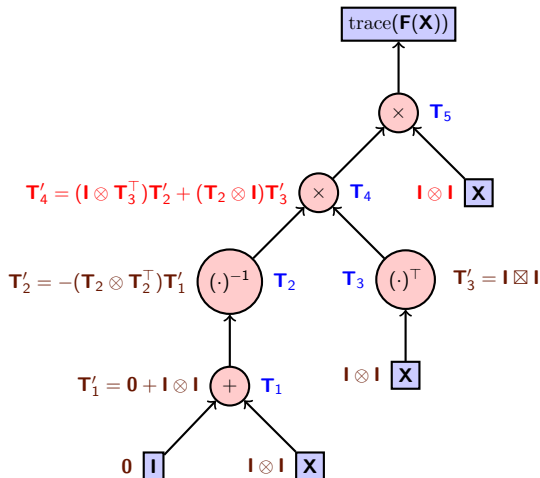
Chain rule: $(\mathbf{T}_1^{-1})' = -(\mathbf{T}_1^{-1} \otimes \mathbf{T}_1^{-\top}) \mathbf{T}_1'$



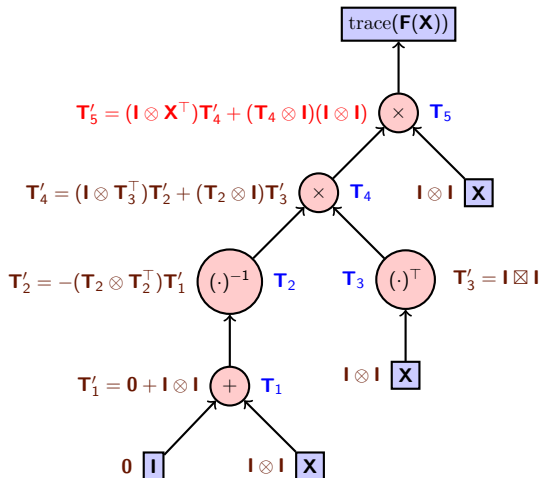
$$(\mathbf{X}^\top)' = \mathbf{I} \boxtimes \mathbf{I}$$



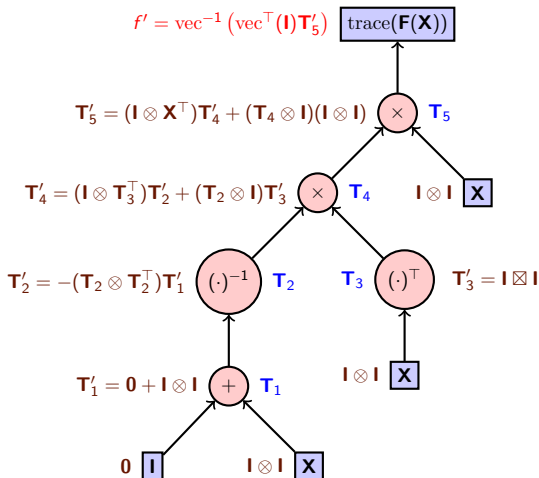
Product rule: $(\mathbf{GH})' = (\mathbf{I} \otimes \mathbf{H}^\top) \mathbf{G}' + (\mathbf{G} \otimes \mathbf{I}) \mathbf{H}'$



Product rule: $(\mathbf{GH})' = (\mathbf{I} \otimes \mathbf{H}^\top) \mathbf{G}' + (\mathbf{G} \otimes \mathbf{I}) \mathbf{H}'$



$$((\text{trace}(\mathbf{F}))')^\top = \text{vec}^{-1}(\text{vec}^\top(\mathbf{I})\mathbf{F}')$$



Forward mode differentiation

- Requires huge intermediate matrices.
- Only the last step reduces the size.

Key points:

- $\mathbf{F}'(\mathbf{X})$ is composed of Kronecker, box, and outer products.
 - Only for a large class of functions.
- These can be “unwound”:

$$\text{vec}^\top(\mathbf{C})\mathbf{A} \otimes \mathbf{B} = \text{vec}^\top(\mathbf{B}^\top \mathbf{C} \mathbf{A})$$

Formation of huge intermediate matrices can be eliminated.

Reverse Mode Differentiation

Transpose

$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$

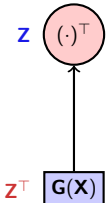


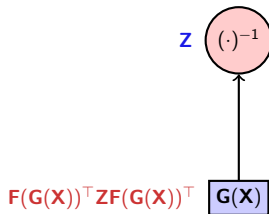
Diagram illustrating the transpose operation: \mathbf{Z}^\top is the output of the transpose operation applied to $\mathbf{G}(\mathbf{X})$.

$$\begin{aligned} \mathbf{F}'(\mathbf{G}(\mathbf{X})) &= (\mathbf{I} \boxtimes \mathbf{I}) \mathbf{G}'(\mathbf{X}); \mathbf{F}(\mathbf{Y}) = \mathbf{Y}^\top \\ \frac{\partial}{\partial \mathbf{X}} \left((\mathbf{G}(\mathbf{X}))^\top \right)^\top \text{vec} \left(\mathbf{Z}^\top \right) &= ((\mathbf{I} \boxtimes \mathbf{I}) \mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\mathbf{Z}^\top \right) \\ &= (\mathbf{G}'(\mathbf{X}))^\top (\mathbf{I} \boxtimes \mathbf{I})^\top \text{vec} \left(\mathbf{Z}^\top \right) \\ &= (\mathbf{G}'(\mathbf{X}))^\top (\mathbf{I} \boxtimes \mathbf{I}) \text{vec} \left(\mathbf{Z}^\top \right) \\ &= (\mathbf{G}'(\mathbf{X}))^\top \text{vec} (\mathbf{Z}) \end{aligned}$$

Reverse Mode Differentiation

Inverse

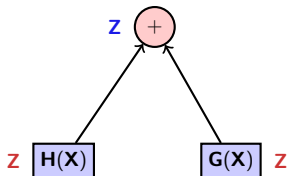
$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$



Reverse Mode Differentiation

Sum

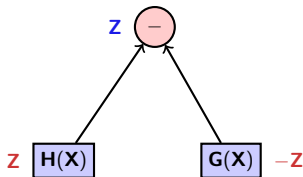
$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$



Reverse Mode Differentiation

Minus

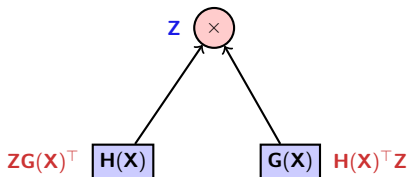
$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} \mathbf{F}(\mathbf{G}(\mathbf{X})) \right)^\top \right)$$



Reverse Mode Differentiation

Product

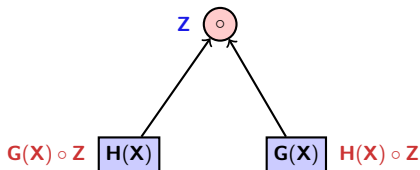
$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$



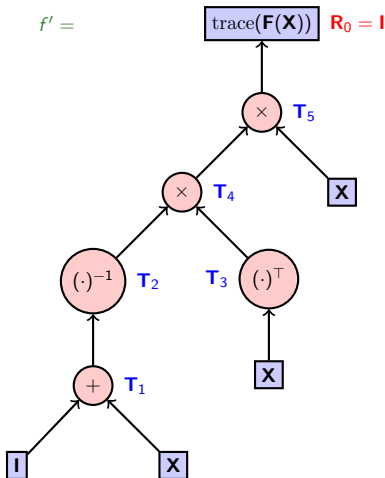
Reverse Mode Differentiation

Element-wise Product

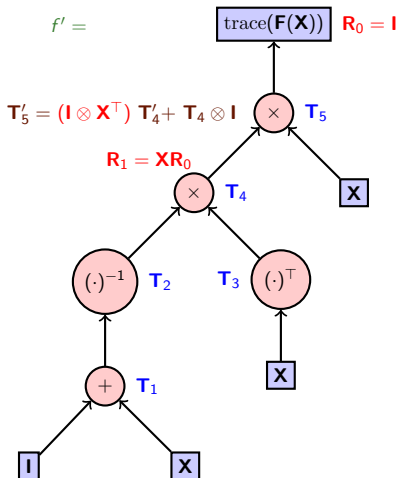
$$\text{vec} \left(\mathbf{F}'(\mathbf{G}(\mathbf{X}))^\top \right) = (\mathbf{G}'(\mathbf{X}))^\top \text{vec} \left(\left(\frac{\partial}{\partial \mathbf{G}(\mathbf{X})} (\mathbf{F}(\mathbf{G}(\mathbf{X}))) \right)^\top \right)$$



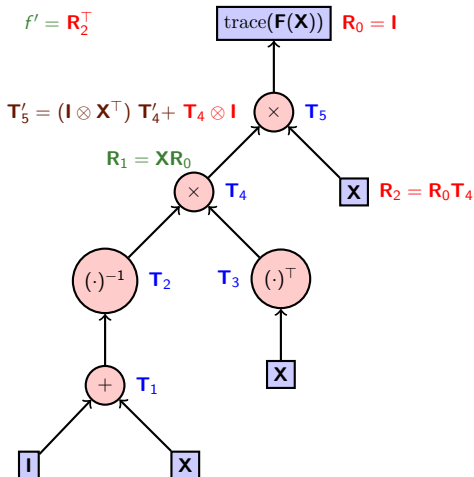
Reverse Mode Differentiation: $(f')^\top = \text{vec}^{-1}(\text{vec}^\top(\mathbf{I})\mathbf{F}')$



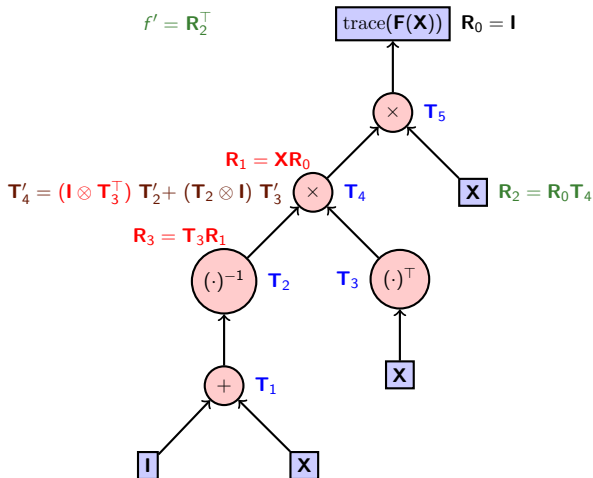
$$\text{vec}^\top(\mathbf{R}_0)(\mathbf{I} \otimes \mathbf{X}^\top) = \text{vec}^\top(\mathbf{X}\mathbf{R}_0\mathbf{I})$$



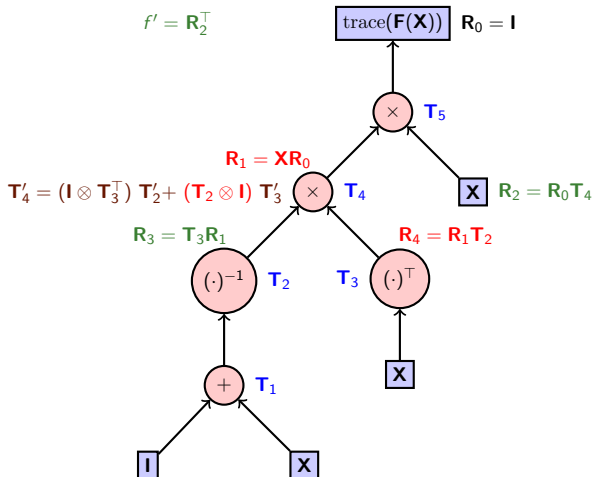
$$\text{vec}^\top(\mathbf{R}_0)(\mathbf{T}_4 \otimes \mathbf{I}) = \text{vec}^\top(\mathbf{I}\mathbf{R}_0\mathbf{T}_4)$$



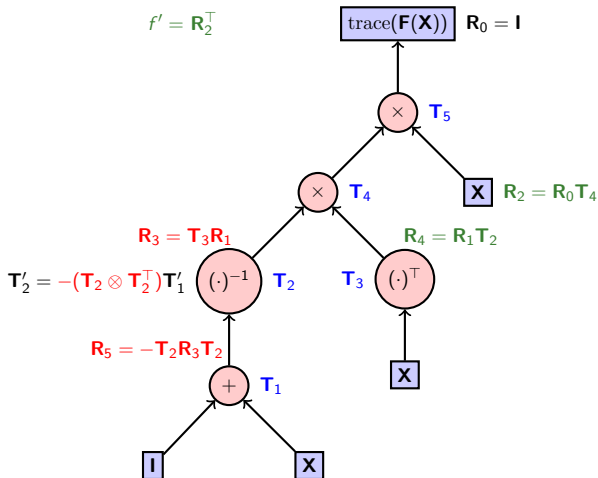
$$\text{vec}^\top(\mathbf{R}_1)(\mathbf{I} \otimes \mathbf{T}_3^\top) = \text{vec}^\top(\mathbf{T}_3 \mathbf{R}_1 \mathbf{I})$$



$$\text{vec}^\top(\mathbf{R}_1)(\mathbf{T}_2 \otimes \mathbf{I}) = \text{vec}^\top(\mathbf{I}\mathbf{R}_1\mathbf{T}_2)$$

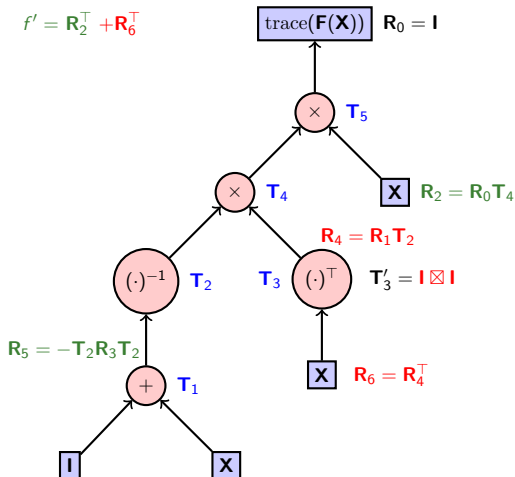


$$\text{vec}^\top(\mathbf{R}_3)(-\mathbf{T}_2 \otimes \mathbf{T}_2^\top) = \text{vec}^\top(-\mathbf{T}_2 \mathbf{R}_3 \mathbf{T}_2)$$



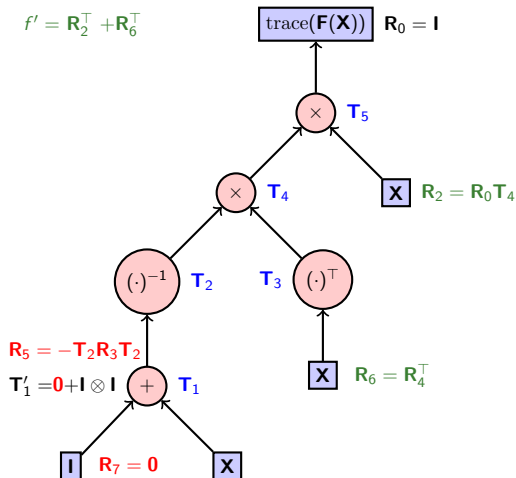
$$\text{vec}^\top(\mathbf{R}_4)(\mathbf{I} \boxtimes \mathbf{I}) = \text{vec}^\top(\mathbf{I} \mathbf{R}_4^\top \mathbf{I})$$

$$f' = \mathbf{R}_2^\top + \mathbf{R}_6^\top$$



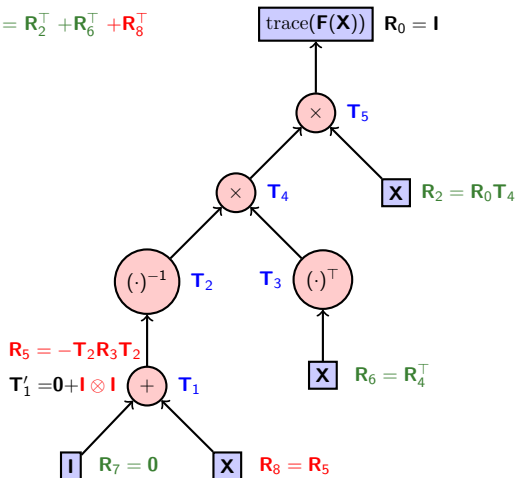
$$\text{vec}^\top(\mathbf{R}_5)\mathbf{0} = \text{vec}(\mathbf{0})$$

$$f' = \mathbf{R}_2^\top + \mathbf{R}_6^\top$$



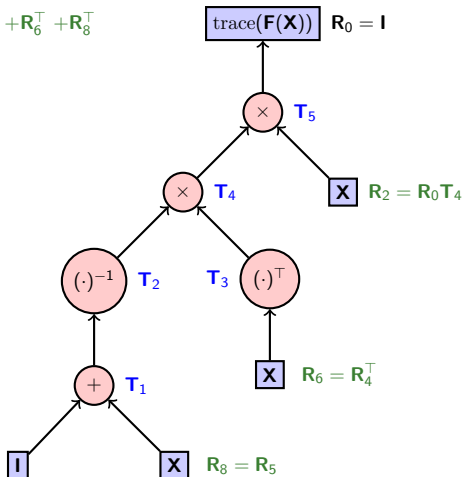
$$\text{vec}^T(\mathbf{R}_5) \mathbf{I} \otimes \mathbf{I} = \text{vec}(\mathbf{R}_5)$$

$$f' = \mathbf{R}_2^T + \mathbf{R}_6^T + \mathbf{R}_8^T$$



$$f'(\mathbf{X}) = \mathbf{R}_2^\top + \mathbf{R}_6^\top + \mathbf{R}_8^\top$$

$$f' = \mathbf{R}_2^\top + \mathbf{R}_6^\top + \mathbf{R}_8^\top$$



Automatic Scalar/Vector Differentiation

Computational Cost of Differentiation

Theorem

The algorithm described for reverse-mode matrix differentiation can be computed with no more than 4 times the arithmetic operations than is needed to compute the function when the matrix operations make no assumptions about the structure of the matrix operands (such as matrices being banded or upper triangular).

- We are working hard to ensure optimizations in our software.
- Non-trivial to carry out all optimizations.

AMD Software

Overview

- Implemented using templates and operator over-loading.
- Provides both **symbolic** and **algorithmic** differentiation.
 - Symbolic differentiation gives results as Matlab code.
 - Algorithmic differentiation:
 - Elemental (dense, multi-threaded, distributed).
 - Eigen (dense and sparse, multi-threaded).
 - Easy to extend to other matrix packages.
- Scalar-matrix: **trace** and **log-determinant**.
- Matrix-matrix: **+**, **-**, *****, **.***, **transpose** and **inverse**.
- Two ways to compute symbolic differentiation:
 - C++ program or on the command line.

AMD Software

Example AMD Program

Define a Scalar-Matrix function `trace(AXBX)`

```
/** Create a variable X and an identity function */  
symbolic_matrix_type X("X", ROW, COL);  
SymbolicMMFunc fX(X, false);  
/** Create constants A,B and corresponding constant functions */  
symbolic_matrix_type A("A", ROW, COL);  
symbolic_matrix_type B("B", ROW, COL);  
SymbolicMMFunc fA(A, true);  
SymbolicMMFunc fB(B, true);  
/** Scalar-matrix function placeholder */  
SymbolicSMFunc func;  
func = trace(fA*fX*fB*fX);  
std::cout << "Function Value: " << func.functionVal.getString();  
std::cout << "Derivative Value: " << func.derivativeVal.getString();
```

AMD Software

Example AMD Program

The resulting derivative is:

```
$ ./SymbolicSample.exe  
Function Value: trace(((A*X)*B)*X)  
Derivative Value: (((B*X)*A)'+((A*X)*B)')
```

OR

Use the symbolic calculator:

```
$ ./SymbolicCalculator.exe 'trace(((A*X)*B)*X)'  
Function Value: trace(((A*X)*B)*X)  
Derivative Value: (((B*X)*A)'+((A*X)*B)')
```

AMD Software

In Progress: Program Optimization

- Common sub-expression elimination.
- Repeated application of matrix and direct-matrix identities.
 - $\mathbf{I} \times \mathbf{X} = \mathbf{X}$, $\mathbf{X} + 0 = \mathbf{X}$, $(\mathbf{A}^\top)^\top = \mathbf{A}$, etc.
 - $(\mathbf{I} \otimes \mathbf{I})\text{vec}(\mathbf{I}) = \mathbf{I}$, $(\mathbf{A} \boxtimes \mathbf{B})^\top = \mathbf{B}^\top \boxtimes \mathbf{A}^\top$, etc.
- Critical in providing performance guarantees.
- Students working on this:
 - Wuwei Zhang (Purdue University)
 - Allan Sepucaia (University of Campinas).

AMD Software

In Progress: Python Interface

```
$ python  
>>>import AMD  
>>>X = AMD.symMat('X')  
>>>fX = AMD.scalarMatrixFunc(X, 1);  
...
```

- Easy to use and prototype.
- Has a large user-base in machine learning.
- Students working on this:
 - Anna Romanova (UCSD)
 - Gabriel Borges (University of Campinas).

Taylor Expansion

Recall the Taylor expansion of a function f around a value x_0 :

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0) + \frac{1}{3!}(x - x_0)^3 f'''(x_0) + \dots$$

For a vector valued function $f'(\mathbf{x}_0)$:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^\top f'(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top f''(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \dots$$

- $f''(\mathbf{x}_0)$ is a matrix.
- $f'''(\mathbf{x}_0)$ is a 3-D tensor.
 - It is unclear how to continue the expansion.

Taylor Expansion

Scalar-matrix Functions: $\text{trace } \mathbf{F}(\mathbf{X})$

Let $\mathbf{F}(\mathbf{X}) = (\mathbf{I} - \mathbf{X})^{-1}$; expanding around $\mathbf{X}_0 = \mathbf{0}$ we get:

$$(\mathbf{I} - \mathbf{X})^{-1} = \mathbf{I} + \mathbf{X} + \mathbf{X}^2 + \mathbf{X}^3 + \dots$$

So, $f(\mathbf{X}) = \text{trace}((\mathbf{I} - \mathbf{X})^{-1})$ we can write the Taylor expansion as:

$$\text{trace}((\mathbf{I} - \mathbf{X})^{-1}) = \text{trace}(\mathbf{I}) + \text{trace}(\mathbf{X}) + \text{trace}(\mathbf{X}^2) + \text{trace}(\mathbf{X}^3) + \dots$$

- We do not compute $f'(\mathbf{X}), f''(\mathbf{X}), f'''(\mathbf{X}), \dots$ explicitly.
- We can compute $f''(\mathbf{X})\text{vec}(\mathbf{X})$ efficiently using our code.
- Automate the process of building the Taylor expansion.

Taylor Expansion

$\log \det(\mathbf{X})$ From Algebra

Let $\mathbf{P} = \mathbf{P}_0 + \mathbf{\Delta}$ be a symmetrix matrix; $\mathbf{X} = \mathbf{P}_0^{-\frac{1}{2}} \mathbf{\Delta} \mathbf{P}_0^{-\frac{1}{2}}$.

$$\begin{aligned} \log \det \mathbf{P} &= \log \det(\mathbf{P}_0 + \mathbf{\Delta}) \\ &= \log \det(\mathbf{P}_0^{1/2} (\mathbf{I} + \mathbf{P}_0^{-1/2} \mathbf{\Delta} \mathbf{P}_0^{-1/2}) \mathbf{P}_0^{1/2}) \\ &= \log \det \mathbf{P}_0 + \log \det(\mathbf{I} + \mathbf{P}_0^{-1/2} \mathbf{\Delta} \mathbf{P}_0^{-1/2}) \\ &= \log \det \mathbf{P}_0 + \log \det(\mathbf{I} + \mathbf{X}) \end{aligned}$$

Taylor Expansion

$\log \det(\mathbf{X})$ From Algebra

Let $\mathbf{P} = \mathbf{P}_0 + \mathbf{\Delta}$ be a symmetric matrix; $\mathbf{X} = \mathbf{P}_0^{-\frac{1}{2}} \mathbf{\Delta} \mathbf{P}_0^{-\frac{1}{2}}$. Let, $\mathbf{Q}^\top \mathbf{E} \mathbf{Q}$ be the eigen-decomposition of \mathbf{X} .

$$\begin{aligned}
 \log \det \mathbf{P} &= \log \det(\mathbf{P}_0 + \mathbf{\Delta}) \\
 &= \log \det \mathbf{P}_0 + \log \det(\mathbf{I} + \mathbf{X}) \\
 &= \log \det \mathbf{P}_0 + \log \det(\mathbf{Q}^\top \mathbf{Q} + \mathbf{Q}^\top \mathbf{E} \mathbf{Q}) \\
 &= \log \det \mathbf{P}_0 + \log \det(\mathbf{Q}^\top (\mathbf{I} + \mathbf{E}) \mathbf{Q}) \\
 &= \log \det \mathbf{P}_0 + \log \det(\mathbf{I} + \mathbf{E}) \\
 &= \log \det \mathbf{P}_0 + \sum_{i=1}^d \log(1 + e_i)
 \end{aligned}$$

Taylor Expansion

$\log \det(\mathbf{X})$ From Algebra

Let $\mathbf{P} = \mathbf{P}_0 + \mathbf{\Delta}$ be a symmetrix matrix; $\mathbf{X} = \mathbf{P}_0^{-\frac{1}{2}} \mathbf{\Delta} \mathbf{P}_0^{-\frac{1}{2}}$. Let, $\mathbf{Q}^\top \mathbf{E} \mathbf{Q}$ be the eigen-decomposition of \mathbf{X} .

$$\begin{aligned} \log \det \mathbf{P} &= \log \det(\mathbf{P}_0 + \mathbf{\Delta}) \\ &= \log \det \mathbf{P}_0 + \sum_{i=1}^d \log(1 + e_i) \\ &= \log \det \mathbf{P}_0 + \sum_{i=1}^d \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} e_i^k \\ &= \log \det \mathbf{P}_0 + \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \sum_{i=1}^d e_i^k \\ &= \log \det \mathbf{P}_0 + \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{trace}(\mathbf{E}^k) \end{aligned}$$

Taylor Expansion

Convenience: $\text{trace}(\mathbf{E}^k) = \text{trace}(\mathbf{X}^k)$:

When, $\mathbf{X} = \mathbf{Q}^\top \mathbf{E} \mathbf{Q}$, we have $\text{trace}(\mathbf{E}^k) = \text{trace}(\mathbf{X}^k)$:

$$\begin{aligned} \text{trace}(\mathbf{E}^k) &= \text{trace}(\mathbf{E} \mathbf{E} \cdots \mathbf{E}) \\ &= \text{trace}(\mathbf{E} \mathbf{Q} \mathbf{Q}^\top \mathbf{E} \mathbf{Q} \mathbf{Q}^\top \cdots \mathbf{E} \mathbf{Q} \mathbf{Q}^\top) \\ &= \text{trace}(\mathbf{Q}^\top \mathbf{E} \mathbf{Q} \mathbf{Q}^\top \mathbf{E} \mathbf{Q} \mathbf{Q}^\top \cdots \mathbf{E} \mathbf{Q}) \\ &= \text{trace}((\mathbf{Q}^\top \mathbf{E} \mathbf{Q})^k) \\ &= \text{trace}(\mathbf{X}^k) \end{aligned}$$

We use the following:

- $\mathbf{Q}^\top \mathbf{Q} = \mathbf{Q} \mathbf{Q}^\top = \mathbf{I}$ when \mathbf{X} is symmetric.
- $\text{trace}(\cdot)$ is invariant under unitary transforms.

Taylor Expansion

Scalar-Matrix Functions: $\log \det \mathbf{P}$

$$\begin{aligned}
 \log \det \mathbf{P} &= \log \det \mathbf{P}_0 + \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{trace}(\mathbf{E}^k) \\
 &= \log \det \mathbf{P}_0 + \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{trace}(\mathbf{X}^k) \\
 &= \log \det(\mathbf{P}_0) + \text{trace}(\mathbf{X}) - 1/2 \text{trace}(\mathbf{X}^2) + \dots \\
 &= \log \det(\mathbf{P}_0) + \text{trace}(\Delta \mathbf{P}_0^{-1}) - 1/2 \text{trace}(\Delta \mathbf{P}_0^{-1} \Delta \mathbf{P}_0^{-1}) + \dots
 \end{aligned}$$

Notice that:

$$\begin{aligned}
 \text{trace}(\Delta \mathbf{P}_0^{-1}) &= \text{trace} \left(\Delta \left(\frac{\partial}{\partial \mathbf{P}_0} (\log \det \mathbf{P}_0) \right)^{\top} \right) \\
 \text{trace}(\Delta \mathbf{P}_0^{-1} \Delta \mathbf{P}_0^{-1}) &= \text{trace} \left(\Delta \left(\frac{\partial}{\partial \mathbf{P}_0} \left(\text{trace}(\Delta \mathbf{P}_0^{-1}) \right) \right)^{\top} \right)
 \end{aligned}$$

Taylor Expansion

$\log \det(\mathbf{X})$

If we define $\Delta = (\mathbf{X} - \mathbf{X}_0)$ to be a constant function (with respect to \mathbf{X}_0) we can compute the Taylor series iteratively as follows:

$$f_0(\mathbf{X}_0) = \log \det(\mathbf{X}_0)$$

$$f_1(\mathbf{X}_0) = \text{trace}(\Delta(f'_0(\mathbf{X}_0))^T)$$

$$f_2(\mathbf{X}_0) = \text{trace}(\Delta(f'_1(\mathbf{X}_0))^T)$$

$$f_3(\mathbf{X}_0) = \text{trace}(\Delta(f'_2(\mathbf{X}_0))^T)$$

The Taylor expansion can now be written:

$$\log \det(\mathbf{X}) = f_0(\mathbf{X}_0) + f_1(\mathbf{X}_0) + \frac{1}{2!} f_2(\mathbf{X}_0) + \frac{1}{3!} f_3(\mathbf{X}_0) \cdots$$

Taylor expansion for $\log \det(\mathbf{X})$ continued

```
/* Initialize matrices */
SymbolicMatrixMatlab X("X",M,M), X0("X0",M,M), Delta("(X-X0)",M,M);
SymbolicScalarMatlab a2("1/2!"), a3("1/3!");
/* Define Scalar-Matrix and Matrix-Matrix functions */
SymbolicSMFunc r2(a2,M,M), r3(a3,M, M);
SymbolicMMFunc fX(X, false), fX0(X0, false), fDelta(Delta, true);
/* Compute Taylor series terms iteratively */
SymbolicSMFunc f0 = logdet(fX0);
SymbolicSMFunc f1 = trace(fDelta * transpose(*f0.derivativeFuncVal));
SymbolicSMFunc f2 = trace(fDelta * transpose(*f1.derivativeFuncVal));
SymbolicSMFunc f3 = trace(fDelta * transpose(*f2.derivativeFuncVal));
/* Taylor Expansion with four terms */
SymbolicSMFunc func = f0 + f1 + r2*f2 + r3*f3;
```


Here is the result:

./TaylorSample.exe

The first 4 terms of Taylor Expansion for $\log \det(X)$ around X_0 is:

```
((log(det(X0))
+trace((X-X0)*inv(X0)))
+(1/2!*trace((X-X0)*(-(inv(X0)*(X-X0))*inv(X0)))))
+(1/3!*trace((X-X0)*( -(inv(X0)*((X-X0)*(inv(X0)*(-(X-X0))))*inv(X0)))'
+(-(inv(X0)*((-X-X0))*(inv(X0)*(X-X0)))*inv(X0)))'))))
```

Conclusion

Thanks For Your Attention