The differential calculus of causal functions

David Sprunger
National Institute of Informatics, Tokyo JP
sprunger@nii.ac.jp

Bart Jacobs
Radboud University, Nijmegen NL
bart@cs.ru.nl

Abstract

Causal functions of sequences occur throughout computer science, from theory to hardware to machine learning. Mealy machines, synchronous digital circuits, signal flow graphs, and recurrent neural networks all have behaviour that can be described by causal functions. In this work, we examine a differential calculus of causal functions which includes many of the familiar properties of standard multivariable differential calculus. These causal functions operate on infinite sequences, but this work gives a different notion of an infinite-dimensional derivative than either the Fréchet or Gateaux derivative used in functional analysis. In addition to showing many standard properties of differentiation, we show causal differentiation obeys a unique recurrence rule. We use this recurrence rule to compute the derivative of a simple recurrent neural network called an Elman network by hand and describe how the computed derivative can be used to train the network.

2012 ACM Subject Classification Mathematics of computing → Differential calculus; Computing methodologies → Neural networks

Keywords and phrases sequences, causal functions, derivatives, recurrent neural networks, Elman networks

Digital Object Identifier 10.4230/LIPIcs.CALCO.2019.23

Funding *David Sprunger*: This author is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

1 Introduction

Many computations on infinite data streams operate in a *causal* manner, meaning their k^{th} output depends only on the first k inputs. Mealy machines, clocked digital circuits, signal flow graphs, recurrent neural networks, and discrete time feedback loops in control theory are a few examples of systems performing such computations. When designing these kinds of systems to fit some specification, a common issue is figuring out how adjusting one part of the system will affect the behaviour of the whole. If the system has some real-valued semantics, as is especially common in machine learning or control theory, the derivative of these semantics with respect to a quantity of interest, say an internal parameter, gives a locally-valid first-order estimate of the system-wide effect of a small change to that quantity. Unfortunately, since the most natural semantics for infinite data streams is in an infinite-dimensional vector space, it is not practical to use the resulting infinite-dimensional derivative.

To get around this, one tactic is to replace the infinite system by a finite system obtained by an approximation or heuristic and take derivatives of the replacement system. This can be seen, for example, in *backpropagation through time* [13], which trains a recurrent neural network by first unrolling the feedback loop the appropriate number of times and then applying traditional backpropagation to the unrolled network.

This tactic has the advantage that we can take derivatives in a familiar (finite-dimensional) setting, but the disadvantage that it is not clear what properties survive the approximation

process from the unfamiliar (infinite-dimensional) setting. For example, it is not immediately clear whether backpropagation through time obeys the usual rules of differential calculus, like a sum or chain rule, nor is this issue confronted in the literature, to the best of our knowledge. Thus, useful compositional properties of differentiation are ignored in exchange for a comfortable setting in which to do calculus.

In this work, we take advantage of the fact that causal functions between sequences are already essentially limits of finite-dimensional functions and therefore have derivatives which can also be expressed as essentially limits of the derivatives of these finite-dimensional functions. This leads us to the basics of a differential calculus of causal functions. Unlike *arbitrary* functions between sequences, this limiting process allows us to avoid the use of normed vector spaces, and so we believe our notion of derivative is distinct from Fréchet derivatives.

Outline. In section 2, we define causal functions and recall several mechanisms by which these functions on infinite data can be defined. In particular, we recall a coalgebraic scheme finding causal functions as the behaviour of Mealy machines (proposition 6), and give a definitional scheme in terms of so-called *finite approximants* (definition 8). In section 3, we define differentiability and derivatives of causal functions on real-vector sequences (definition 12) and compute several examples. In section 4, we obtain several rules for our differential causal calculus analogous to those of multivariable calculus, including a chain rule, parallel rule, sum rule, product rule, reciprocal rule, and quotient rule (propositions 18, 19, 22, 23, 26, and 27, respectively). We additionally find a new rule without a traditional analogue we call the recurrence rule (theorem 28). Finally, in section 5, we apply this calculus to find derivatives of a simple kind of recurrent neural network called an Elman network [6] by hand. We also demonstrate how to use the derivative of the network with respect to a parameter to guide updates of that parameter to drive the network towards a desired behaviour.

2 Causal functions of sequences

A *sequence* or *stream* in a set A is a countably infinite list of values from A, which we also think of as a function from the natural numbers ω to A. If σ is a stream in A, we denote its value at $k \in \omega$ by σ_k. We may also think of a stream as a listing of its image, like $\sigma = (\sigma_0, \sigma_1, \ldots)$. The set of all sequences in A is denoted A^ω.

Given $a \in A$ and $\sigma \in A^\omega$, we can form a new sequence by prepending a to σ. The sequence $a : \sigma$ is defined by $(a : \sigma)_0 = a$ and $(a : \sigma)_{k+1} = \sigma_k$. This operation can be extended to prepend arbitrary finite-length words $w \in A^*$ by the obvious recursion. Conversely, we can destruct a given sequence into an element and a second sequence with functions $\texttt{hd} : A^\omega \to A$ and $\texttt{tl} : A^\omega \to A^\omega$ defined by $\texttt{hd}(\sigma) = \sigma_0$ and $\texttt{tl}(\sigma)_k = \sigma_{k+1}$.

▶ **Definition 1** (slicing). *If $\sigma \in A^\omega$ is a stream and $j \leq k$ are natural numbers, the* slicing *$\sigma_{j:k}$ is the list $(\sigma_j, \sigma_{j+1}, \ldots, \sigma_k) \in A^{k-j+1}$.*

▶ **Definition 2** (causal function). *A function $f : A^\omega \to B^\omega$ is* causal *means $\sigma_{0:k} = \tau_{0:k}$ implies $f(\sigma)_{0:k} = f(\tau)_{0:k}$ for all $\sigma, \tau \in A^\omega$ and $k \in \omega$.*

2.1 Causal functions via coalgebraic finality

A standard coalgebraic approach to causal functions is to view them as the behaviour of Mealy machines.

▶ **Definition 3** (Mealy functor). *Given two sets A, B, the functor $M_{A,B} : \mathbf{Set} \to \mathbf{Set}$ is defined by $M_{A,B}(X) = (B \times X)^A$ on objects and $M_{A,B}(f) : \phi \mapsto (\mathrm{id}_B \times f) \circ \phi$ on morphisms.*

$M_{A,B}$-coalgebras are Mealy machines with input alphabet A and output alphabet B, and possibly an infinite state space. The set of causal functions $A^\omega \to B^\omega$ carries a final $M_{A,B}$-coalgebra using the following operations, originally observed by Rutten in [10].

▶ **Definition 4.** *The* Mealy output *of a causal function $f : A^\omega \to B^\omega$ is the function $\mathtt{hd}f : A \to B$ defined by $(\mathtt{hd}f)(a) = f(a : \sigma)_0$ for any $\sigma \in A^\omega$.*

▶ **Definition 5.** *Given $a \in A$ and a causal function $f : A^\omega \to B^\omega$, the* Mealy ($a$-)derivative *of f is the causal function $\partial_a f : A^\omega \to B^\omega$ defined by $(\partial_a f)(\sigma) = \mathtt{tl}(f(a : \sigma))$.*

Note $\mathtt{hd}(f)$ is well-defined even though σ may be freely chosen due to the causality of f.

▶ **Proposition 6** (Proposition 2.2, [10])**.** *The set of causal functions $A^\omega \to B^\omega$ carries an $M_{A,B}$-coalgebra via $f \mapsto \lambda a.((\mathtt{hd}f)(a), \partial_a f)$, which is a final $M_{A,B}$-coalgebra.*

Hence, a coalgebraic methodology for defining causal functions is to define a Mealy machine and take the image of a particular state in the final coalgebra. By constructing the Mealy machine cleverly, one can ensure the resulting causal function has some desired properties. This is the core idea behind the "syntactic method" using GSOS definitions in [8]. In that work, a Mealy machine of terms is built in such a way that all causal functions $(A^k)^\omega \to A^\omega$ can be recovered.

▶ **Example 7.** Suppose $(A, +_A, \cdot_A, 0_A)$ is a vector space over $\mathbb{R}$. This vector space structure can be extended to A^ω componentwise in the obvious way. To illustrate the coalgebraic method, we characterise this structure with coalgebraic definitions.

To define sequence vector sum coalgebraically, we define a Mealy machine $1 \to (A \times 1)^{A \times A}$ with one state, satisfying $\mathtt{hd}(s)(a, a') = a +_A a'$ and $\partial_{(a,a')}(s) = s$. Then $+_{A^\omega} : (A \times A)^\omega \to A^\omega$ is defined to be the image of s in the final $M_{A^2,A}$-coalgebra.

Note that technically the vector sum in A^ω should be a function of type $A^\omega \times A^\omega \to A^\omega$, so we are tacitly using the isomorphism between $(A \times A)^\omega$ and $A^\omega \times A^\omega$. We will be using similar recastings of sequences in the sequel without bringing up this point again.

The zero vector can similarly be defined by a single state Mealy machine $1 \to (A \times 1)^1$ with input alphabet 1 and output alphabet A, satisfying $\mathtt{hd}(s')(*) = 0_A$ and $\partial_*(s') = s'$. The zero vector of A^ω is the global element picked out by the image of s'.

Finally, scalar multiplication can be defined with a Mealy machine $\mathbb{R} \to (A \times \mathbb{R})^A$ with states $r \in \mathbb{R}$, such that $\mathtt{hd}(r)(a) = r \cdot_A a$ and $\partial_a r = r$. Then $r \cdot_{A^\omega} \sigma \triangleq [\![r]\!](\sigma)$, where $[\![r]\!]$ is the image of r in the final $M_{A,A}$-coalgebra.

We immediately begin dropping the subscripts from $+_{A^\omega}$ and $\cdot_{A^\omega}$ and when the relevant vector space can be inferred from context.

2.2 Causal functions via finite approximation

Another approach to causal functions is consider them as a limit of finite approximations, replacing the single function on infinite data with infinitely many functions on finite data. There are (at least) two approaches with this general style, which we briefly describe next.

▶ **Definition 8.** *Let $f : A^\omega \to B^\omega$ be a causal function and $\sigma \in A^\omega$.*

The pointwise approximation *of f is the sequence of functions $U_k(f) : A^{k+1} \to B$ defined by $U_k(f)(w) \triangleq f(w : \sigma)_k$.*

The stringwise approximation *of f is the sequence of functions $T_k(f) : A^{k+1} \to B^{k+1}$ defined by $T_k(f)(w) \triangleq f(w : \sigma)_{0:k}$.*

Again, these are well-defined despite σ being arbitrary due to f's causality. We chose the letters U and T deliberately—sometimes the pointwise approximants of a causal function are called its Unrollings, and the stringwise approximants are called its Truncations.

Conversely, given an arbitrary collection of functions $u_k : A^{k+1} \to B$ for $k \in \omega$, there is a unique causal function whose pointwise approximation is the sequence u_k. Thus we have the following bijective correspondence:

$$\frac{A^\omega \longrightarrow B^\omega \text{ causal}}{A^{k+1} \longrightarrow B \text{ for each } k \in \omega} \tag{1}$$

We can nearly do the same for stringwise approximations, but the sequence $t_k : A^{k+1} \to B^{k+1}$ must satisfy $t_k(w) = t_{k+1}(wa)_{0:k}$ for all $w \in A^{k+1}$ and $a \in A$.

The interchangeability between a causal function and its approximants is a crucial theme in this work. Since a function's pointwise and stringwise approximants are inter-obtainable, we will sometimes refer to a causal function's "finite approximants" by which we mean either family of approximants.

2.3 Causal functions via recurrence

Finite approximants are a very flexible way of defining causal functions, but causal functions may have a more compact representation when they conform to a regular pattern. Recurrence is one such pattern where a causal function is defined by repeatedly using an ordinary function $g : A \times B \to B$ and an initial value $i \in B$ to obtain $\mathtt{rec}_i(g) : A^\omega \to B^\omega$ via:

$$[\mathtt{rec}_i(g)(\sigma)]_k = \begin{cases} g(\sigma_0, i) & \text{if } k = 0 \\ g(\sigma_k, [\mathtt{rec}_i(g)(\sigma)]_{k-1}) & \text{if } k > 0 \end{cases}$$

Recurrent definitions can be converted into finite approximant definitions using the following: $U_k(\mathtt{rec}_i(g))(\sigma_{0:k}) = g(\sigma_k, g(\sigma_{k-1}, \ldots g(\sigma_1, g(\sigma_0, i)) \ldots))$. Note these pointwise approximants satisfy the recurrence relation $U_k(\mathtt{rec}_i(g))(\sigma_{0:k}) = g(\sigma_k, U_{k-1}(\mathtt{rec}_i(g))(\sigma_{0:k-1}))$.

▶ **Example 9.** The unary running product function $\prod : \mathbb{R}^\omega \to \mathbb{R}^\omega$ can be defined by a recurrence relation:

$$\prod(\sigma) = \tau \Leftrightarrow \left\{ \tau_{k+1} = \sigma_{k+1} \cdot \tau_k \quad \text{after } \tau_0 = \sigma_0 \cdot 1 \right.$$

Here g is multiplication of reals and $i = 1$. In approximant form, $[\prod(\sigma)]_k = \prod_{i=0}^{k} \sigma_i$.

A special case of recurrent causal functions occurs when there is an $h : A \to B$ such that $g(a, b) = h(a)$ for all $(a, b) \in A \times B$. In this case, $[\mathtt{rec}_i(g)(\sigma)]_k = h(\sigma_k)$ and in particular does not depend on the initial value i or any entry σ_j for $j < k$. We denote $\mathtt{rec}_i(g)$ by $\mathtt{map}(h)$ in this special case since it maps h componentwise across the input sequence.

3 Differentiating causal functions

Our goal in this work is to develop a basic differential calculus for causal functions. Thus we will focus our attention on causal functions between real-vector sequences $(\mathbb{R}^n)^\omega$ for $n \in \omega$, specializing from causal functions on general sets from the last section. We will draw many of our illustrating examples for derivatives from Rutten's *stream calculus* [9], which describes many such causal functions between real-number streams. More importantly, [9] establishes many useful algebraic properties of these functions rigorously via coalgebraic methods.

There are many different approaches one might consider to defining differentiable causal functions. One might be to take the original coalgebraic definition and replace the underlying category (**Set**) with a category of finite-dimensional Cartesian spaces and differentiable (or smooth) maps. Unfortunately, the space of differentiable functions between finite-dimensional spaces is not finite-dimensional, so the exponential needed to define the $M_{A,B}$ functor in this category does not exist.

Another approach is to think of causal functions as functions between infinite dimensional vector spaces and take standard notions from analysis, like Fréchet derivatives, and apply them in this context. However, norms on sequence spaces usually impose a finiteness condition like bounded or square-summable on the domains and ranges of sequence functions. These restrictions are compatible with many causal functions like the pointwise sum function above, but other causal functions like the running product function become significantly less interesting.

Our approach to differentiating causal functions is to consider a causal function differentiable when all of its finite approximants are differentiable via the correspondence (1). We will develop this idea rigorously in section 3.2, but first we need to know a bit about linear causal functions.

3.1 Linear causal functions

Stated abstractly, the derivative of a function at a point is a linear map which provides an approximate change in the output of a function given an input representing a small change in the input to that function [11]. Since linear functions $\mathbb{R} \to \mathbb{R}$ are in bijective correspondence with their slopes, typically in single-variable calculus the derivative of a function at a point is instead given as a single real number. In multivariable calculus, derivatives are usually represented by (Jacobian) matrices since matrices represent linear maps between finite dimensional spaces. Linear functions between infinite dimensional vector spaces do not have a similarly compact, computationally-useful representation, but we can still define derivatives of (causal) functions at points to be linear (causal) maps.

We described the natural vector space structure of $(\mathbb{R}^n)^\omega$ in Example 7. A linear causal function is a causal function which is also linear with respect to this vector space structure.

▶ **Definition 10.** *A causal function $f : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ is* linear *when $f(r \cdot \sigma) = r \cdot f(\sigma)$ and $f(\sigma + \tau) = f(\sigma) + f(\tau)$ for all $r \in \mathbb{R}$ and $\sigma, \tau \in (\mathbb{R}^n)^\omega$.*

▶ **Lemma 11.** *Let $f : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ be a causal function. The following are equivalent:*

1. *f is linear,*
2. *$U_k(f) : (\mathbb{R}^n)^{k+1} \to \mathbb{R}^m$ is linear for all $k \in \omega$, and*
3. *$T_k(f) : (\mathbb{R}^n)^{k+1} \to (\mathbb{R}^m)^{k+1}$ is linear for all $k \in \omega$.*

This refines the correspondence (1), allowing us to define a linear causal function by naming linear finite approximants.

Since linear functions between finite dimensional vector spaces can be represented by matrices, we can think of linear causal functions as limits of the matrices representing its finite approximants. This view results in row-finite infinite matrices, such as:

$$\begin{bmatrix} A_{00} & 0 & 0 & \dots \\ A_{10} & A_{11} & 0 & \dots \\ A_{20} & A_{21} & A_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

where the A_{ij} are m-row, n-column blocks such that for $j > i$ all entries are 0. These are related to the matrices for the approximants of the causal function as follows.

1. The matrix $\begin{bmatrix} A_{k0} & A_{k1} & \dots & A_{kk} \end{bmatrix}$ is the matrix representing $U_k(f)$.
2. The matrix $\begin{bmatrix} A_{00} & 0 & 0 & \dots & 0 \\ A_{10} & A_{11} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{k0} & A_{k1} & A_{k2} & \dots & A_{kk} \end{bmatrix}$ is the matrix representing $T_k(f)$. The compatibility conditions on the functions $T_k(f)$ ensure that the matrix for $T_k(f)$ can be found in the upper left corner of the matrix for $T_{k+1}(f)$. Note also the upper triangular nature of the matrices for $T_k(f)$ are a consequence of causality—the first m outputs can depend only on the first n inputs, so the last entries in the top row must all be 0 and so on.

Unlike finite-dimensional matrices, we do not think these infinite matrices are a computationally useful representation, but they are conceptually useful to get an idea of how causal linear functions can be considered the limit of their linear truncations.

3.2 Definition of derivative

As we have mentioned, we will use the derivatives of the approximants of a causal function to define the derivative of the causal function itself. We denote the m-row, n-column Jacobian matrix of a differentiable function $\varphi : \mathbb{R}^n \to \mathbb{R}^m$ at $x \in \mathbb{R}^n$ by $J\varphi(x)$. Recall this matrix is

$$\begin{bmatrix} \frac{\partial \varphi_1}{\partial x_1}(x) & \frac{\partial \varphi_1}{\partial x_2}(x) & \dots & \frac{\partial \varphi_1}{\partial x_n}(x) \\ \frac{\partial \varphi_2}{\partial x_1}(x) & \frac{\partial \varphi_2}{\partial x_2}(x) & \dots & \frac{\partial \varphi_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varphi_m}{\partial x_1}(x) & \frac{\partial \varphi_m}{\partial x_2}(x) & \dots & \frac{\partial \varphi_m}{\partial x_n}(x) \end{bmatrix}$$

where $\varphi_i : \mathbb{R}^n \to \mathbb{R}$ and $\varphi = \langle \varphi_1, \dots \varphi_m \rangle$. We will also be glossing over the distinction between a matrix and the linear function it represents, using $J\varphi(x)$ to mean either when convenient.

▶ **Definition 12.** *A causal function $f : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ is* differentiable at $\sigma \in (\mathbb{R}^n)^\omega$ *if all of its finite approximants $U_k(f) : (\mathbb{R}^n)^{k+1} \to \mathbb{R}^m$ are differentiable at $\sigma_{0:k}$ for all $k \in \omega$. If f is differentiable at σ, the* derivative of f at σ *is the unique linear causal function $\mathcal{D}^* f(\sigma) : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ satisfying $U_k(\mathcal{D}^* f(\sigma)) = J(U_k(f))(\sigma_{0:k})$.*

In this definition we are using the correspondence (1), refined in Lemma 11, which allows us to define a causal (linear) function by specifying its (linear) finite approximants. We could equally well have used stringwise approximants in this definition rather than pointwise approximants, as the following lemma states.

▶ **Lemma 13.** *The causal function f is differentiable at σ if and only if each of $T_k(f)$ are differentiable at $\sigma_{0:k}$ for all $k \in \omega$. In this case, $\mathcal{D}^* f(\sigma)$ satisfies $T_k(\mathcal{D}^* f(\sigma)) = J(T_k(f))(\sigma_{0:k})$.*

Though we have mentioned this is not particularly useful computationally, the derivative of a differentiable function at a point has a representation as a row-finite infinite matrix.

▶ **Lemma 14.** *If f is differentiable at σ, each $U_k(f) : (\mathbb{R}^n)^{k+1} \to \mathbb{R}^m$ has an m-row, $n(k+1)$-column Jacobian matrix representing its derivative at $\sigma_{0:k}$. Let A_{ki} be m-row, n-column*

blocks of this Jacobian, so that $J(U_k(f))(\sigma_{0:k}) = \begin{bmatrix} A_{k0} & A_{k1} & \dots & A_{kk} \end{bmatrix}$ *The derivative of* f *at* σ *is the linear causal function represented by the row-finite infinite matrix*

$$\mathcal{D}^* f(\sigma) = \begin{bmatrix} A_{00} & 0 & 0 & \dots \\ A_{10} & A_{11} & 0 & \dots \\ A_{20} & A_{21} & A_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Note that this linear causal function can be evaluated at a sequence $\Delta\sigma \in (\mathbb{R}^n)^\omega$ by multiplying the infinite matrix by $\Delta\sigma$, considered as an infinite column vector.

3.3 Examples

Next, we use this definition of derivative to find the causal derivatives of some basic functions from Rutten's stream calculus.

▶ **Example 15.** We show the pointwise sum stream function $+ : (\mathbb{R}^2)^\omega \to \mathbb{R}^\omega$ is its own derivative at every point $(\sigma, \tau) \in (\mathbb{R}^2)^\omega$. Note $U_k(+)(\sigma_0, \tau_0, \dots, \sigma_k, \tau_k) = \sigma_k + \tau_k$, so $J(U_k(+))(\sigma_0, \tau_0, \dots, \sigma_k, \tau_k) = \begin{bmatrix} 0 & \dots & 0 & 1 & 1 \end{bmatrix}$. This is the matrix representation of $U_k(+)$ itself, so $(\mathcal{D}^*+)(\sigma, \tau) = +$ or, in other notation, $(\mathcal{D}^*+)(\sigma, \tau)(\Delta\sigma, \Delta\tau) = \Delta\sigma + \Delta\tau$ for any $\sigma, \tau, \Delta\sigma, \Delta\tau \in \mathbb{R}^\omega$.

This argument can be repeated for all pointwise sum functions $+ : (\mathbb{R}^n \times \mathbb{R}^n)^\omega \to (\mathbb{R}^n)^\omega$, replacing the "1" blocks in the Jacobian above with I_n.

Since the derivative of any constant $x : 1 \to \mathbb{R}^n$ is $0_{\mathbb{R}^n} : 1 \to \mathbb{R}^n$, the derivative of any constant sequence must necessarily be the zero sequence. In stream calculus, there are two important constant sequences defined corecursively: $[r]$ defined by $\mathtt{hd}([r])(*) = r$ and $\partial_*([r]) = [0]$ for all $r \in \mathbb{R}$ and X defined by $\mathtt{hd}(X)(*) = 0$ and $\partial_*(X) = [1]$. Written out as sequences, $[r] = (r, 0, 0, 0, \dots)$ and $X = (0, 1, 0, 0, \dots)$.

▶ **Example 16.** $\mathcal{D}^*[r] = \mathcal{D}^* X = [0]$.

Next, we consider the Cauchy sequence product. Under the correspondence between sequences $\sigma \in \mathbb{R}^\omega$ and formal power series $\sum \sigma_i x^i \in \mathbb{R}[[x]]$, the Cauchy product is the sequence operation corresponding to the (Cauchy) product of formal power series. This operation is coalgebraically characterized in Rutten [9] as the unique function $\times : (\mathbb{R}^2)^\omega \to \mathbb{R}^\omega$ satisfying $\mathtt{hd}(\times)(s_0, t_0) = s_0 \cdot t_0$ and $(\partial_{(s_0,t_0)}\times)(\sigma, \tau) = \mathtt{tl}(\sigma) \times \tau + [s_0] \times \mathtt{tl}(\tau)$. For our purposes, the explicit definition is more useful: $U_k(\times)(\sigma_{0:k}, \tau_{0:k}) = \sum_{i=0}^k \sigma_i \cdot \tau_{k-i}$.

▶ **Example 17.** We compute the derivative of the Cauchy product.

$$J(U_k(\times))(\sigma_0, \tau_0, \dots, \sigma_k, \tau_k) = \begin{bmatrix} \tau_k & \sigma_k & \tau_{k-1} & \sigma_{k-1} & \dots & \tau_0 & \sigma_0 \end{bmatrix}$$

Notice that multiplying this matrix by (an initial segment) of a small change sequence $(\Delta\sigma_0, \Delta\tau_0, \dots, \Delta\sigma_k, \Delta\tau_k)$ yields

$$J(U_k(\times))(\sigma_0, \tau_0, \dots, \sigma_k, \tau_k)(\Delta\sigma_0, \Delta\tau_0, \dots, \Delta\sigma_k, \Delta\tau_k) = \sum_{i=0}^k \Delta\sigma_i \cdot \tau_{k-i} + \sum_{i=0}^k \sigma_i \cdot \Delta\tau_{k-i}$$

Therefore, $(\mathcal{D}^*\times(\sigma, \tau))(\Delta\sigma, \Delta\tau) = \Delta\sigma \times \tau + \sigma \times \Delta\tau$.

Another sequence product considered in the stream calculus is the Hadamard product, also called the pointwise product. Defined coalgebraically, the Hadamard product is the unique binary operation defined by $\mathtt{hd}(\odot)(s_0, t_0) = s_0 \cdot t_0$ and $(\partial_{(s_0,t_0)}\odot)(\sigma,\tau) = \mathtt{tl}(\sigma)\odot\mathtt{tl}(\tau)$. This has a similar derivative to the Cauchy product: $\mathcal{D}^*\odot(\sigma,\tau)(\Delta\sigma,\Delta\tau) = \Delta\sigma\odot\tau + \sigma\odot\Delta\tau$.

Note that these derivatives make sense without any reference to properties of the sequences used. We are not aware of a way to realize this derivative as an instance of a notion of derivative known in analysis. The most obvious notion to try is a Fréchet derivative induced by a norm on the space of sequences. However, all norms we know on these spaces, including ℓ^p-norms and γ-geometric norms $\|\sigma\| = \sum \sigma_i \cdot \gamma^i$ for $\gamma \in (0, 1]$, restrict the space of sequences to various extents.

4 Rules of causal differentiation

Just as it is impractical to compute all derivatives from the definition in undergraduate calculus, it is also impractical to compute causal derivatives directly from the definition. To ease this burden, one typically proves various "rules" of differentiation which provide compositional recipes for finding derivatives. That is our task in this section.

There are at least two good reasons to hope *a priori* that the standard rules of differentiation might hold for causal derivatives. First, causal derivatives were defined to agree with standard derivatives in their finite approximants. Since these approximant derivatives satisfy these rules, we might hope that they hold over the limiting process. Second, **smooth** causal functions form a Cartesian differential category, as was shown in [12]. The theory of Cartesian differential categories includes as axioms or theorems abstract versions of the chain rule, sum rule, etc. However, neither of these reasons are immediately sufficient, so we must provide independent justification.

4.1 Basic rules and their consequences

We begin by stating some rules familiar from undergraduate calculus.

▶ **Proposition 18** (causal chain rule). *Suppose $f : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ and $g : (\mathbb{R}^m)^\omega \to (\mathbb{R}^\ell)^\omega$ are causal functions. Suppose further f is differentiable at $\sigma \in (\mathbb{R}^n)^\omega$ and g is differentiable at $f(\sigma)$. Then $h = g \circ f$ is differentiable at σ and its derivative is $\mathcal{D}^*g(f(\sigma)) \circ \mathcal{D}^*f(\sigma)$.*

Proof. Let $f_k = T_k(f)$, $g_k = T_k(g)$, and $h_k = T_k(h)$. We know $h_k = g_k \circ f_k$. We show the stringwise approximants of $\mathcal{D}^*(g \circ f)(\sigma)$ and $\mathcal{D}^*g(f(\sigma)) \circ \mathcal{D}^*f(\sigma)$ match.

$$\begin{aligned} T_k(\mathcal{D}^*(g\circ f)(\sigma)) &= J(h_k)(\sigma_{0:k}) = J(g_k \circ f_k)(\sigma_{0:k}) \\ &= J(g_k)(f_k(\sigma_{0:k})) \times J(f_k)(\sigma_{0:k}) \qquad (*) \\ &= J(g_k)(f(\sigma)_{0:k}) \times J(f_k)(\sigma_{0:k}) \\ &= T_k(\mathcal{D}^*g(f(\sigma))) \circ T_k(\mathcal{D}^*f(\sigma)) = T_k(\mathcal{D}^*g(f(\sigma)) \circ \mathcal{D}^*f(\sigma)) \end{aligned}$$

where the starred line is by the classical chain rule. ◀

Since we have already overloaded $\times$ for both Cauchy stream product and matrix product, we use $\|$ for the parallel composition of functions, where the parallel composition of $\phi : \mathbb{R}^n \to \mathbb{R}^m$ and $\psi : \mathbb{R}^p \to \mathbb{R}^q$ is $\phi\|\psi : \mathbb{R}^{n+p} \to \mathbb{R}^{m+q}$ defined by $(\phi\|\psi)(x, y) = (\phi(x), \psi(y))$ for $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^p$. We do not know of a standard name for this rule, but in multivariable calculus there is a rule $J(\phi\|\psi)(x, y) = J\phi(x)\|J\psi(y)$, which we shall call the parallel rule. There is a similar rule for causal derivatives we describe next.

▶ **Proposition 19** (causal parallel rule). *Suppose $f : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ and $h : (\mathbb{R}^p)^\omega \to (\mathbb{R}^q)^\omega$ are causal functions, and that they are differentiable at $\sigma \in (\mathbb{R}^n)^\omega$ and $\tau \in (\mathbb{R}^p)^\omega$, respectively. Then $f\|h : (\mathbb{R}^{n+p})^\omega \to (\mathbb{R}^{m+q})^\omega$ is differentiable at $(\sigma, \tau) \in (\mathbb{R}^{n+p})^\omega$ and its derivative is $\mathcal{D}^* f(\sigma)\|\mathcal{D}^* h(\tau)$.*

Proof. The stringwise approximants of $\mathcal{D}^*(f\|h)(\sigma,\tau)$ and $\mathcal{D}^* f(\sigma)\|\mathcal{D}^* h(\tau)$ match:

$$\begin{aligned} T_k(\mathcal{D}^*(f\|h)(\sigma,\tau)) &= J(T_k(f\|h))(\sigma_{0:k},\tau_{0:k}) = J(T_k(f)\|T_k(h))(\sigma_{0:k},\tau_{0:k}) \\ &= J(T_k(f))(\sigma_{0:k})\|J(T_k(h))(\tau_{0:k}) \qquad (*) \\ &= T_k(\mathcal{D}^* f(\sigma))\|T_k(\mathcal{D}^* h(\tau)) = T_k(\mathcal{D}^* f(\sigma)\|\mathcal{D}^* h(\tau)) \end{aligned}$$

where the starred line is by the classical parallel rule. ◀

▶ **Proposition 20** (causal linearity). *If $f : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ is a linear causal function, it is differentiable at every $\sigma \in (\mathbb{R}^n)^\omega$ and its derivative is $\mathcal{D}^* f(\sigma) = f$.*

These three results are the fundamental properties of causal differentiation we will be using. Many other standard rules are consequences of these. For example, we can derive a sum rule from these properties.

▶ **Definition 21.** *The* sum *of two causal maps $f, g : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ is defined to be $f + g \triangleq + \circ (f\|g) \circ \Delta_{(\mathbb{R}^n)^\omega}$, where $\Delta_{(\mathbb{R}^n)^\omega}$ is the sequence duplication map.*

▶ **Proposition 22** (causal sum rule). *If f and g as in Definition 21 are both differentiable at σ, so is their sum and its derivative is $\mathcal{D}^* f(\sigma) + \mathcal{D}^* g(\sigma)$.*

Proof. Using the properties above, we find

$$\begin{aligned} \mathcal{D}^*(f+g)(\sigma) &= \mathcal{D}^*(+ \circ (f\|g) \circ \Delta_{(\mathbb{R}^n)^\omega})(\sigma) && \text{(sum of maps def'n)} \\ &= \mathcal{D}^*(+)((f\|g \circ \Delta_{(\mathbb{R}^n)^\omega})(\sigma)) \circ \mathcal{D}^*(f\|g \circ \Delta_{(\mathbb{R}^n)^\omega})(\sigma) && \text{(causal chain rule)} \\ &= + \circ \mathcal{D}^*(f\|g \circ \Delta_{(\mathbb{R}^n)^\omega})(\sigma) && \text{(linearity of +)} \\ &= + \circ \mathcal{D}^*(f\|g)(\Delta_{(\mathbb{R}^n)^\omega}(\sigma)) \circ \mathcal{D}^*(\Delta_{(\mathbb{R}^n)^\omega})(\sigma) && \text{(causal chain rule)} \\ &= + \circ \mathcal{D}^*(f\|g)(\sigma,\sigma) \circ \Delta_{(\mathbb{R}^n)^\omega} && \text{(def'n \& linearity of } \Delta) \\ &= + \circ (\mathcal{D}^* f(\sigma)\|\mathcal{D}^* g(\sigma)) \circ \Delta_{(\mathbb{R}^n)^\omega} && \text{(causal parallel rule)} \\ &= \mathcal{D}^* f(\sigma) + \mathcal{D}^* g(\sigma) && \text{(sum of maps def'n)} \end{aligned}$$

as desired. ◀

For functions $f, g : \mathbb{R}^\omega \to \mathbb{R}^\omega$, we can define their Cauchy and Hadamard products $f \times g$ and $f \odot g$ with the pattern of Definition 21 and prove two product rules using the derivatives of the binary operations $\times$ and $\odot$ we computed earlier.

▶ **Proposition 23** (causal product rules). *If $f, g : \mathbb{R}^\omega \to \mathbb{R}^\omega$ are causal functions differentiable at σ, so are their Cauchy and Hadamard products, and their derivatives are*

$$\mathcal{D}^*(f \times g)(\sigma)(\Delta\sigma) = \mathcal{D}^* f(\sigma)(\Delta\sigma) \times g(\sigma) + f(\sigma) \times \mathcal{D}^* g(\sigma)(\Delta\sigma)$$
$$\mathcal{D}^*(f \odot g)(\sigma)(\Delta\sigma) = \mathcal{D}^* f(\sigma)(\Delta\sigma) \odot g(\sigma) + f(\sigma) \odot \mathcal{D}^* g(\sigma)(\Delta\sigma)$$

A typical point of confusion in undergraduate calculus is the role of constants: sometimes they are treated like elements of the underlying vector space and sometimes like functions which always return that vector. In our calculus, a constant can similarly sometimes mean a fixed sequence picked out by $c : 1 \to (\mathbb{R}^n)^\omega$ or the composition of this map after a discarding map $!_{(\mathbb{R}^n)^\omega} : (\mathbb{R}^n)^\omega \to 1$. We have described the derivative of a constant element in Example 16, now we treat constant maps.

▶ **Proposition 24** (causal constant rule). *The derivative of* $!_{(\mathbb{R}^n)^\omega} : (\mathbb{R}^n)^\omega \to 1$ *is* $!_{(\mathbb{R}^n)^\omega}$*. If* $c : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ *is a constant map, its derivative is the constant map* $[0](\sigma) \equiv 0_{(\mathbb{R}^m)^\omega}$*.*

▶ **Proposition 25** (causal constant multiple rule). *If* $c : \mathbb{R}^\omega \to \mathbb{R}^\omega$ *is a constant function and* $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ *is any other causal function differentiable at* σ*, so is* $c \times f$ *and its derivative is* $c \times \mathcal{D}^* f(\sigma)$*.*

Proof. Combine the causal product rule and the causal constant rule. ◀

4.2 Implicit causal differentiation

We have seen the standard rules presented in the last section are useful as computational shortcuts, just as they are in undergraduate calculus. In the causal calculus they turn out to be perhaps even more crucial, since some differentiable causal functions do not have simple closed forms, so trying to find their derivative from the definition is extremely difficult.

The *stream inverse* [9] is the first partial causal function we will consider. This operation is defined on $\sigma \in \mathbb{R}^\omega$ such that $\sigma_0 \neq 0$ with the unbounded-order recurrence relation

$$[\sigma^{-1}]_k = \begin{cases} \frac{1}{\sigma_0} & \text{if } k = 0 \\ -\frac{1}{\sigma_0} \cdot \sum_{i=0}^{k-1} \left(\sigma_{n-i} \cdot [\sigma^{-1}]_i\right) & \text{if } k > 0 \end{cases}.$$

Reasoning about this function in terms of its components is extraordinarily difficult since each component is defined in terms of all the preceding components. However, there is a useful fact from Rutten [9] which we can use to find the derivative of this operation at all σ where it is defined: $\sigma \times \sigma^{-1} = [1]$.

▶ **Proposition 26** (causal reciprocal rule). *The partial function* $(\cdot)^{-1} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ *is differentiable at all* $\sigma \in \mathbb{R}^\omega$ *such that* $\sigma_0 \neq 0$*, and its derivative is*

$$(\mathcal{D}^*(\cdot)^{-1})(\sigma)(\Delta\sigma) = [-1] \times \sigma^{-1} \times \sigma^{-1} \times \Delta\sigma$$

Proof. Since $\sigma \times \sigma^{-1} = [1]$, their derivatives must also be equal. In particular:

$$[0] = \mathcal{D}^*[1] = \mathcal{D}^*(\sigma \times \sigma^{-1})(\Delta\sigma) = \sigma \times (\mathcal{D}^*(\cdot)^{-1})(\sigma)(\Delta\sigma) + \Delta\sigma \times (\sigma^{-1})$$

using the causal product rule. Solving this equation for $(\mathcal{D}^*(\cdot)^{-1})(\sigma)(\Delta\sigma)$ yields

$$(\mathcal{D}^*(\cdot)^{-1})(\sigma)(\Delta\sigma) = [-1] \times \sigma^{-1} \times \sigma^{-1} \times \Delta\sigma$$

where we are implicitly using many of the identities established in [9]. ◀

When adopting the conventions that $\sigma^{-n} \triangleq \sigma^{-(n-1)} \times \sigma^{-1}$ and $\sigma \times \tau^{-1} \triangleq \frac{\sigma}{\tau}$, this rule looks quite like the usual rule for the derivative of the reciprocal function: $(J(\cdot)^{-1})(x)(\Delta x) = -\frac{\Delta x}{x^2}$.

▶ **Proposition 27** (causal quotient rule). *If* $f, g : \mathbb{R}^\omega \to \mathbb{R}^\omega$ *are causal functions differentiable at* σ *and* $g(\sigma)_0 \neq 0$*, then* $\frac{f}{g}$ *is also differentiable at* σ *and its derivative is*

$$\frac{\mathcal{D}^* f(\sigma)(\Delta\sigma) \times g(\sigma) + [-1] \times f(\sigma) \times \mathcal{D}^* g(\sigma)(\Delta\sigma)}{g(\sigma)^2}.$$

4.3 The recurrence rule

So far, causal differential calculus is rather similar to traditional differential calculus. There are two different product rules corresponding to two different products. We were forced to use an implicit differentiation trick to find the derivative of the reciprocal function, but in the end we found a familiar result. However, next we state a rule with no traditional analogue.

▶ **Theorem 28** (causal recurrence rule). *Let $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ be differentiable (everywhere) and $i \in \mathbb{R}^m$. Then $\mathtt{rec}_i(g) : (\mathbb{R}^n)^\omega \to (\mathbb{R}^m)^\omega$ is differentiable (everywhere) as a causal function and its derivative $\Delta\tau \triangleq [\mathcal{D}^*\mathtt{rec}_i(g)](\sigma)(\Delta\sigma)$ satisfies the following recurrence:*

$$\begin{cases} \tau_{k+1} = g(\sigma_{k+1}, \tau_k) & \text{after } \tau_0 = g(\sigma_0, i) \\ \Delta\tau_{k+1} = Jg(\sigma_{k+1}, \tau_k)(\Delta\sigma_{k+1}, \Delta\tau_k) & \text{after } \Delta\tau_0 = Jg(\sigma_0, i)(\Delta\sigma_0, 0_{\mathbb{R}^m}) \end{cases}$$

Proof. We check $U_k(\mathcal{D}^*\mathtt{rec}_i(g)(\sigma))(\Delta\sigma_{0:k}) = \Delta\tau_k$ by induction on k. To simplify our notation, we write $u_k \triangleq U_k(\mathtt{rec}_i(g))$. The base case is easy:

$$\begin{aligned} U_0([\mathcal{D}^*\mathtt{rec}_i(g)](\sigma))(\Delta\sigma_0) &= J(U_0(\mathtt{rec}_i(g)))(\sigma_0)(\Delta\sigma_0) \\ &= J(\lambda x.g(x,i))(\sigma_0)(\Delta\sigma_0) = Jg(\sigma_0, i)(\Delta\sigma_0, 0_{\mathbb{R}^m}) \end{aligned}$$

The induction step uses the fact that $u_k(\sigma_{0:k}) = g(\sigma_k, u_{k-1}(\sigma_{0:k-1}))$.

$$\begin{aligned} U_k([\mathcal{D}^*\mathtt{rec}_i(g)](\sigma))(\Delta\sigma_{0:k}) &= Ju_k(\sigma_{0:k})(\Delta\sigma_{0:k}) \\ &= [Jg(\sigma_k, \tau_{k-1}) \circ \langle J\pi_k(\sigma_{0:k}), J(u_{k-1} \circ \overline{\pi_k})(\sigma_{0:k})\rangle](\Delta\sigma_{0:k}) \\ &= [Jg(\sigma_k, \tau_{k-1}) \circ \langle \pi_k, Ju_{k-1}(\sigma_{0:k-1}) \circ \overline{\pi_k}\rangle](\Delta\sigma_{0:k}) \\ &= Jg(\sigma_k, \tau_{k-1})(\Delta\sigma_k, Ju_{k-1}(\sigma_{0:k-1})(\Delta\sigma_{0:k-1})) \\ &= Jg(\sigma_k, \tau_{k-1})(\Delta\sigma_k, \Delta\tau_{k-1}) \end{aligned}$$

where $\overline{\pi_k}$ is the map discarding the last element of a list. ◀

Degenerate recurrences, which do not refer to previous values generated by the recurrence, are a special instance of this rule.

▶ **Corollary 29** (causal map rule). *Let $h : \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable function. Then $\mathtt{map}(h)$ is differentiable as a causal function, and its derivative is $\mathtt{map}(Jh)$.*

To illustrate the recurrence rule, we revisit the running product function, introduced in Example 9, and compute its derivative.

▶ **Example 30.** The unary running product function $\prod : \mathbb{R}^\omega \to \mathbb{R}^\omega$ was defined to be $\mathtt{rec}_1(g)$ where g is binary multiplication of reals. In approximant form, $U_k(\mathtt{rec}_1(g))(\sigma_{0:k}) = \prod_{i=0}^k \sigma_i$. We compute a recurrence for the derivative of this function using the recurrence rule.

Since g is binary multiplication, $Jg(s,t)(\Delta s, \Delta t) = \Delta s \cdot t + s \cdot \Delta t$. By the recurrence rule, $[\mathcal{D}^*\mathtt{rec}_i(g)](\sigma)(\Delta\sigma)$ satisfies the recurrence

$$\begin{cases} \tau_{k+1} = \sigma_{k+1} \cdot \tau_k & \text{after } \tau_0 = \sigma_0 \\ \Delta\tau_{k+1} = \Delta\sigma_{k+1} \cdot \tau_k + \sigma_{k+1} \cdot \Delta\tau_k & \text{after } \Delta\tau_0 = \Delta\sigma_0 \end{cases}$$

Note that a direct computation of the derivative of this function is available since we have a simple form for its pointwise approximants. Directly from the definition we would get

$$\Delta\tau_k = U_k(\mathcal{D}^*\mathtt{rec}_1(g)(\sigma))(\Delta\sigma_{0:k}) = \sum_{i=0}^k \prod_{j=0}^k \rho_{ij}$$

where ρ_{ij} is σ_j if $i \neq j$ and $\Delta\sigma_j$ otherwise.

Used naively, this formula results in $O(k^2)$ real number multiplications, and requires access to the entire initial segment of σ at all times. In contrast, computing the same quantity using the recurrence obtained by the recurrence rule requires $O(k)$ multiplications and can be computed on-the-fly, requiring only the availability of the first elements of σ and $\Delta\sigma$ to make initial progress and releasing their memory just after use.

5 An extended example: Elman networks

We next turn toward a potential application domain of our causal differential calculus: machine learning. In particular, we demonstrate that it is possible to use this calculus in the training of recurrent neural networks (RNNs). RNNs differ from the more common feedforward network in that they are designed to process sequences of inputs rather than single inputs. This makes them especially useful in analyzing long texts (sequences of words), spoken language (sequences of sounds), and videos (sequences of images). In fact, particular RNN architectures are the core underlying technologies of many speech recognition products today, such as Alexa and Siri.

In this section, we will be using our causal differential calculus to find the derivative of a simple kind of recurrent neural network, namely an Elman network [6]. This is an influential early example of a network with feedback, though modern feedback networks typically have more structure. Elman networks can operate on sequences of vectors from $\mathbb{R}^n$, but to keep things slightly simpler we will consider Elman networks operating on sequences of real numbers only.

Let $\alpha, \beta, \gamma, \delta, \epsilon \in \mathbb{R}$ be arbitrary parameters and $\phi_1, \phi_2 : \mathbb{R} \to \mathbb{R}$ be arbitrary differentiable "activation" functions.[1] Given an input sequence $\sigma \in \mathbb{R}^\omega$, the Elman network defined by these parameters produces the sequence $E(\sigma) = \tau \in \mathbb{R}^\omega$ satisfying the following recurrence:

$$\begin{cases} \rho_{k+1} = \phi_1(\alpha\sigma_{k+1} + \beta\rho_k + \gamma) & \text{after } \rho_0 = \phi_1(\alpha\sigma_0 + \gamma) \\ \tau_{k+1} = \phi_2(\delta\rho_{k+1} + \epsilon) & \text{after } \tau_0 = \phi_2(\delta\rho_0 + \epsilon) \end{cases}$$

In our notation, if we define $g_1(x, y) \triangleq \phi_1(\alpha x + \beta y + \gamma)$ and $g_2(x) \triangleq \phi_2(\delta x + \epsilon)$, then $E \triangleq \mathtt{map}(g_2) \circ \mathtt{rec}_0(g_1)$. We can therefore find the causal derivative of this Elman network relatively easily using the causal chain rule and causal recurrence rule. Indeed, letting $\mathcal{D}^* E(\sigma)(\Delta\sigma) = \Delta\tau$, these rules tell us $\Delta\tau$ satisfies the recurrence:

$$\begin{cases} \rho_{k+1} = \phi_1(\alpha\sigma_{k+1} + \beta\rho_k + \gamma) & \text{after } \rho_0 = \phi_1(\alpha\sigma_0 + \gamma) \\ \tau_{k+1} = \phi_2(\delta\rho_{k+1} + \epsilon) & \text{after } \tau_0 = \phi_2(\delta\rho_0 + \epsilon) \\ \Delta\rho_{k+1} = \phi_1'(\alpha\sigma_{k+1} + \beta\rho_k + \gamma) \cdot (\alpha\Delta\sigma_{k+1} + \beta\Delta\rho_k) & \text{after } \Delta\rho_0 = \phi_1'(\alpha\sigma_0 + \gamma) \cdot (\alpha\Delta\sigma_0) \\ \Delta\tau_{k+1} = \phi_2'(\delta\rho_{k+1} + \epsilon) \cdot (\delta\Delta\rho_{k+1}) & \text{after } \Delta\tau_0 = \phi_2'(\delta\rho_0 + \epsilon) \cdot (\delta\Delta\rho_0) \end{cases}$$

This derivative tells us how we would expect the output of the Elman network to change in response to a small change $\Delta\sigma$ to its input sequence σ. This can be useful information in analyzing the behavior of the network. However, we can also use causal differentiation to predict how the network's output would change in response to a small change in one of the *parameters*, which is a crucial piece of information used when training the network.

[1] "Activation" here has no technical meaning, but carries a connotation that the function is likely taken from a folklore set of functions including the sigmoid function, hyperbolic tangent, softplus, rectified linear unit, and logistic function. Usually these functions have bounded range, often $[0, 1]$.

Let us now imagine that we have some data on how this Elman network *should* behave, in the form of an input/output pair $(\hat{\sigma}, \hat{\tau}) \in \mathbb{R}^\omega \times \mathbb{R}^\omega$ representing ground truth, and we want to figure out how to adjust one of the parameters, say α, so that our Elman network better reflects this ground truth.

We can define a causal function related to the Elman network E, but where we now consider α to be a variable and fix σ to be $\hat{\sigma}$. Denote this function $E_{\hat{\sigma}} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ and note that if $\tau = E_{\hat{\sigma}}(\hat{\alpha})$ for $\hat{\alpha} \in \mathbb{R}^\omega$, then τ satisfies the recurrence relation

$$\begin{cases} \rho_{k+1} = \phi_1(\alpha\hat{\sigma}_{k+1} + \beta\rho_k + \gamma) & \text{after } \rho_0 = \phi_1(\alpha\hat{\sigma}_0 + \gamma) \\ \tau_{k+1} = \phi_2(\delta\rho_{k+1} + \epsilon) & \text{after } \tau_0 = \phi_2(\delta\rho_0 + \epsilon) \end{cases}$$

We have simplified our expression using the fact that parameters are fixed values that do not change in the course of the computation of the output sequence, so $\hat{\alpha}_k = \alpha$ for all $k \in \omega$. Similarly, when we make small change to this parameter, that small change will remain independent of the entry in the sequence, so $\widehat{\Delta\alpha}_k = \Delta\alpha$ for all k.

We can compute the derivative of this recurrence relation similarly to above, and find it will satisfy the following recurrence relation:

$$\begin{cases} \rho_{k+1} = \phi_1(\alpha\hat{\sigma}_{k+1} + \beta\rho_k + \gamma) & \text{after } \rho_0 = \phi_1(\alpha\hat{\sigma}_0 + \gamma) \\ \tau_{k+1} = \phi_2(\delta\rho_{k+1} + \epsilon) & \text{after } \tau_0 = \phi_2(\delta\rho_0 + \epsilon) \\ \Delta\rho_{k+1} = \phi_1'(\alpha\hat{\sigma}_{k+1} + \beta\rho_k + \gamma) \cdot (\Delta\alpha\hat{\sigma}_{k+1} + \beta\Delta\rho_k) & \text{after } \Delta\rho_0 = \phi_1'(\alpha\hat{\sigma}_0 + \gamma) \cdot (\Delta\alpha\hat{\sigma}_0) \\ \Delta\tau_{k+1} = \phi_2'(\delta\rho_{k+1} + \epsilon) \cdot (\delta\Delta\rho_{k+1}) & \text{after } \Delta\tau_0 = \phi_2'(\delta\rho_0 + \epsilon) \cdot (\delta\Delta\rho_0) \end{cases}$$

▶ **Example 31.** Let us take a very specific example to illustrate this process. We instantiate the above Elman network with $\alpha = \beta = \delta = 1$, $\gamma = 0.1$, $\epsilon = -0.1$ and $\phi_1 = \phi_2$ are both the sigmoid function.[2]

We suppose our ground truth data tells us a sequence starting $\hat{\sigma} = (1, 1, 1, 1, \ldots)$ should be sent to a sequence starting $\hat{\tau} = (0.60, 0.63, 0.63, 0.64, \ldots)$. In reality, our Elman network as currently parametrized sends $\hat{\sigma}$ to $(0.65707, 0.68226, 0.68503, 0.68533, \ldots)$, when rounded to 5 decimal places. Our task is to decide how to adjust α so that the new network will better match our data, in particular reducing every entry by about 0.05.

We begin by first writing out the recurrence relation for the derivative of $E_{\hat{\sigma}}$ from above with our particular choice of parameters. Since we have chosen many coefficients and all the entries of $\hat{\sigma}$ to be 1, there is significant simplification:

$$\begin{cases} \rho_{k+1} = \phi(\rho_k + 1.1) & \text{after } \rho_0 = \phi(1.1) \\ \tau_{k+1} = \phi(\rho_{k+1} - 0.1) & \text{after } \tau_0 = \phi(\rho_0 - 0.1) \\ \Delta\rho_{k+1} = \phi'(\rho_k + 1.1) \cdot (\Delta\alpha + \Delta\rho_k) & \text{after } \Delta\rho_0 = \phi'(1.1) \cdot \Delta\alpha \\ \Delta\tau_{k+1} = \phi'(\rho_{k+1} - 0.1) \cdot \Delta\rho_{k+1} & \text{after } \Delta\tau_0 = \phi'(\rho_0 - 0.1) \cdot \Delta\rho_0 \end{cases}$$

The only free variable in this recurrence is $\Delta\alpha$. We choose $\Delta\alpha = 0.1$, for reasons to be explained later. Then we can compute $\Delta\tau = (0.00422, 0.00302, 0.00265, 0.00259, \ldots)$.

What does this tell us? The recurrence is supposed to compute the derivative of $E_{\hat{\sigma}}$ at 1 and apply the resulting linear map to 0.1. Using the interpretation of derivative as approximate change, this suggests that if we increase our parameter α from its current value of 1 by $\Delta\alpha =$

[2] The sigmoid function $\phi : \mathbb{R} \to \mathbb{R}$ is defined by $\phi(x) = \frac{1}{1+e^{-x}}$. The sigmoid function is traditionally denoted by σ, but since we have been using σ as a sequence variable we use ϕ.

0.1, we should expect $E_{\hat{\sigma}}(1.1)$ to be about $E_{\hat{\sigma}}(1) + (0.00422, 0.00302, 0.00265, 0.00259, \ldots)$. Since our goal is to **reduce** the output of the network, this adjustment is not a great idea.

What are we to do? One option is to pick a new value for $\Delta\alpha$ and recompute the approximate change, but there is a smarter way. We know that the derivative of $E_{\hat{\sigma}}$ at 1 is linear, so if we instead *decrease* α by 0.1, we would expect $E_{\hat{\sigma}}(0.9)$ to be about $E_{\hat{\sigma}}(1) - (0.00422, 0.00302, 0.00265, 0.00259, \ldots) = (0.65285, 0.67923, 0.68238, 0.68274, \ldots)$. Indeed, after making this adjustment, we find $E_{\hat{\sigma}}(0.9) = (0.65273, 0.67908, 0.68224, 0.68261, \ldots)$. This adjustment ended up decreasing the result by about 0.00015 more than we predicted, which amounts to approximately a 5% overshot of the original prediction.

While it is nice to know our prediction about the change was fairly accurate, subtracting 0.1 from α has not achieved our goal: in each component, our Elman network's output decreased by at most 0.005 while we were trying to create a reduction of 0.05. A natural idea here would be to *really* exploit the linearity of the derivative and make a bigger adjustment to α, namely subtracting $\frac{0.05}{0.005} \cdot \Delta\alpha = 10 \cdot \Delta\alpha = 1$. Computing $E_{\hat{\sigma}}(0)$, we find it is actually $(0.60467, 0.63445, 0.64095, 0.64235, \ldots)$, which is much closer to our goal than $E_{\hat{\sigma}}(0.9)$ turned out to be.

This seems like good news, but if we check the accuracy of the prediction our derivative makes, we would find that the actual reduction from $E_{\hat{\sigma}}(1)$ to $E_{\hat{\sigma}}(0)$ is between 25% and 65% greater than the derivative predicted. Thus, though we were able to make greater progress aligning our network with ground truth, the bigger adjustment came with much greater error. This is a classic tradeoff in neural network training: the linear approximation provided by the derivative is only valid locally, so taking bigger steps along the gradient comes with potentially greater rewards in terms of improvements in network performance but also carries extra risk that greater error could lead the training astray.

6 Conclusion, related work, and future directions

In this paper, we presented a basic differential calculus for causal functions between sequences of real-valued vectors. We gave a definition of derivative for causal functions, showed how to compute derivatives from this definition, established many classical rules from multivariable calculus including the chain, parallel, sum, product, reciprocal, and quotient rules. We additionally showed a rule unique to the causal calculus: the recurrence rule. We then showed how to use these rules in a practical example, namely the training of an Elman network.

Related work. We are not aware of other works directly treating differentiation of causal functions, though we suspect there may be connections to hard-core analysis literature. This work is obviously inspired in results and structure by standard undergraduate multivariable calculus, e.g. [11]. We also have a related categorical treatment of differentiation of causal functions [12] using the framework of Cartesian differential categories [2]. That is much more abstract than the present work, but when concretized to the current scenario would only apply to smooth causal functions.

Though we drew our example differentiable functions almost exclusively from Rutten's stream calculus [9], we would also like to point out signal flow graphs as another interesting treatment of causal functions. an interesting graphical representation of causal functions, investigated in e.g. [1, 3, 4, 7]. We expect that interpreting our differential calculus in this setting could yield a treatment of differentiation in string diagrams.

We suspect recurrence rule we obtained, particularly when differentiating Elman networks, may also have connections to the automatic differentiation literature we are not aware of at this time. In particular, it does rather seem like the recurrence rule augments a recurrence

with dual numbers.

Future directions. As neural networks become more advanced and practitioners find new and interesting ways of using gradients of these networks, we believe theoreticians have a role to play in systematizing the theory of these new applications of derivatives. We believe that the coalgebra community, as experts with many tools for understanding programs operating on, infinite data structures, are particularly well-positioned to help develop these theories. For example, nearly every rule of causal differentiation we established here relies on a coalgebraically-derived property from Rutten's stream calculus [9]. We looked at functions on sequences in particular, but we have every reason to believe further results are possible for more advanced neural network architectures on more exotic infinite data structures.

We are particularly interested in merging our results here with a line of research initiated in [12] using Cartesian differential categories. We believe this causal calculus could be an instance of a Cartesian differential *restriction* category [5], which would drastically improve the scope of our previous results to cover partial and non-smooth causal functions.

References

1. Henning Basold, Marcello Bonsangue, Helle Hvid Hansen, and Jan Rutten. *(Co)Algebraic Characterizations of Signal Flow Graphs*, pages 124–145. Springer International Publishing, Cham, 2014. URL: https://doi.org/10.1007/978-3-319-06880-0_6, doi:10.1007/978-3-319-06880-0_6.
2. R F Blute, J R B Cockett, and R A G Seely. Cartesian differential categories. *Theory and Applications of Categories*, 22:622–672, 2009.
3. Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A categorical semantics of signal flow graphs. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 435–450, 2014. URL: https://doi.org/10.1007/978-3-662-44584-6_30, doi:10.1007/978-3-662-44584-6_30.
4. Filippo Bonchi, Pawel Sobociński, and Fabio Zanasi. Full abstraction for signal flow graphs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 515–526, 2015. URL: https://doi.org/10.1145/2676726.2676993, doi:10.1145/2676726.2676993.
5. JRB Cockett, GSH Cruttwell, and JD Gallagher. Differential restriction categories. *Theory and Applications of Categories*, 25(21):537–613, 2011.
6. Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, Mar 1990. doi:10.1207/s15516709cog1402_1.
7. Stefan Milius. A sound and complete calculus for finite stream circuits. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 421–430, 2010. URL: https://doi.org/10.1109/LICS.2010.11, doi:10.1109/LICS.2010.11.
8. Jan Rutten, Clemens Kupke, and Helle Hvid Hansen. Stream differential equations: Specification formats and solution methods. *Logical Methods in Computer Science*, 13, 2017.
9. J.J.M.M. Rutten. A coinductive calculus of streams. *Mathematical Structures in Computer Science*, 15(1):93–147, Feb 2005. doi:10.1017/S0960129504004517.
10. J.J.M.M. Rutten. Algebraic specification and coalgebraic synthesis of mealy automata. *Electronic Notes in Theoretical Computer Science*, 160:305–319, Aug 2006. doi:10.1016/j.entcs.2006.05.030.
11. Michael Spivak. Calculus on manifolds. 1965.
12. David Sprunger and Shin-ya Katsumata. Differentiable causal computations via delayed trace. *CoRR*, abs/1903.01093, 2019. URL: http://arxiv.org/abs/1903.01093, arXiv:1903.01093.
13. P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990. doi:10.1109/5.58337.