DISCRETE AND CONTINUOUS
DYNAMICAL SYSTEMS
Supplement 2013

Website: www.aimSciences.org

pp. **587–596**

EFFICIENT RECURRENCE RELATIONS FOR UNIVARIATE AND MULTIVARIATE TAYLOR SERIES COEFFICIENTS

Richard D. Neidinger

Davidson College
Box 7002
Davidson, NC 28035-7002, USA

Abstract. The efficient use of Taylor series depends, not on symbolic differentiation, but on a standard set of recurrence formulas for each of the elementary functions and operations. These relationships are often rediscovered and restated, usually in a piecemeal fashion. We seek to provide a fairly thorough and unified exposition of efficient recurrence relations in both univariate and multivariate settings. Explicit formulas all stem from the fact that multiplication of functions corresponds to a Cauchy product of series coefficients, which is more efficient than the Leibniz rule for nth-order derivatives. This principle is applied to function relationships of the form h'=v*u', where the prime indicates a derivative or partial derivative. Each standard (calculator button) function corresponds to an equation, or pair of equations, of this form. A geometric description of the multivariate operation helps clarify and streamline the computation for each desired multi-indexed coefficient. Several research communities use such recurrences including the Differential Transform Method to solve differential equations with initial conditions.

1. **Introduction.** Taylor polynomials for function expressions, and functions given by differential equations, are most efficiently computed by recurrence relations that capture how to combine series. Incomplete glimpses of this idea are encountered in calculus, where a series for e^{x^2} *about 0* is found by substituting into the known series instead of differentiating; in series solutions for *simple* ordinary differential equations (ODEs) where the goal is a recurrence relation; and in numerical ODE solutions where a higher-order Taylor series method is considered as a *theoretical* possibility. A complete set of recurrence relations for operations and standard functions can change the symbolic parts of these methods into practical computational algorithms. Several current research communities use series recurrences as a primary tool: Automatic Differentiation (AD) seeks derivatives or series for functions specified by a computer program (see [4], [8]); Polynomial Methods for Differential Equations was a special session of the conference of this proceedings (e.g. [9]); and the Differential Transform Method is a perspective on directly transforming an ODE (and sometimes a partial differential equation) into a recurrence relation found in some publications (e.g. [5] and the 1986 Chinese textbook referenced therein). Many of the works within these communities recreate or cite piecemeal results for

2010 *Mathematics Subject Classification.* Primary: 65D25, 41A58; Secondary: 41A63, 65L05, 65D15.

Key words and phrases. Series, automatic differentiation, recurrence, multivariable series, differential transform method.

operations on series. We seek a fairly thorough and unified treatment of both univariate and multivariate recurrence relations for such operations.

The core ideas of series recurrence developed over centuries. In 1247, a Chinese (root-finding) algorithm converted coefficients of a polynomial about 0 to coefficients about x_0 [10]. Euler (1707 – 1783) was a master at manipulating series [3]. Franz Mertens (1840 – 1926) proved convergence of the Cauchy product of series [2]. In 1962, the Ph.D. thesis of R.E. Moore used series recurrences (equivalent to many of the univariate rules herein) for error analysis in digital computing [6]. Still, a fairly complete set of univariate rules with derivations is rare [11] and multivariate rules are very hard to find, except possibly as code within specific computing languages [7]. This mathematical presentation is rigorous and self-contained. Although the univariate is a special case of the multivariate, they are treated in succession so that algorithmic ideas for both can be developed simply and then subtle features of the multivariate case can be distinguished.

2. **Univariate recurrences.**

For every function $h : \mathbb{R} \to \mathbb{R}$ analytic in a neighborhood of a, we will associate the Taylor coefficient function $H : \mathbb{N}_0 \to \mathbb{R}$ such that

$$h(x) = \sum_{k=0}^{\infty} H(k)(x-a)^k. \tag{1}$$

The lower case and upper case notation will be used consistently. In the other direction,

$$H(k) = \frac{1}{k!}\left(\frac{d}{dx}\right)^k h(x)\Big|_{x=a}. \tag{2}$$

This can be thought of as a transform method, where (2) is the Taylor Coefficient Transform (or Differential Transform) and (1) is the inverse transform. Actually, $H(k)$ values are rarely computed by differentiating as in (2); instead, special cases are tabulated and combinations or recurrences are used on the transform side. Specifically, Taylor coefficients of elementary functions can be built starting from only the two most basic cases:

- If $h(x) = c \in \mathbb{R}$, then $H(0) = c$ and $H(k) = 0$ otherwise.
- If $h(x) = x$, then $H(0) = a$, $H(1) = 1$ and $H(k) = 0$ otherwise.

Suppose the transforms $U(k)$ and $V(k)$ are already known (as in the above two cases) or have been computed for $u(x) = \sum_{k=0}^{\infty} U(k)(x-a)^k$ and $v(x) = \sum_{k=0}^{\infty} V(k)(x-a)^k$, at least up to some degree $k \le m$. If $h(x)$ is some function or operation on u and/or v, we seek $H(k)$ in terms of U, V and/or preceding values of H. The nature of U, V and H depends on perspective. We will be specifying these by recurrence formulas. In practical application, computers form arrays of floating point doubles $U = (U(0), U(1), U(2), \ldots, U(m))$, $V = (V(0), V(1), V(2), \ldots, V(m))$ and $H = (H(0), H(1), H(2), \ldots, H(m))$. In object-oriented programming, these can be instances of a class (such as the series class in [8]) and operations or standard functions applied to such objects can implement the recurrence formulas to produce the resulting Taylor coefficients. This ability to custom define the action of usual operator symbols or function names, such as $*$ and sin, is called overloading. We now seek recurrence relations for each of the arithmetic operations and standard (calculator button) functions.

2.1. **Arithmetic operations.** The Taylor coefficient transform is linear, by either (1) or (2). If $h(x) = \alpha\, u(x) + \beta\, v(x)$, then $H = \alpha U + \beta V$, as in vector algebra.

The transform of a **product** is the Cauchy product of transforms: If $h(x) = u(x)\, v(x)$, then

$$\begin{aligned} h(x) &= \left(\sum_{j=0}^{\infty} U(j)(x-a)^j\right)\left(\sum_{i=0}^{\infty} V(i)(x-a)^i\right) \\ &= \sum_{k=0}^{\infty}\left(\sum_{j=0}^{k} U(j)V(k-j)\right)(x-a)^k. \end{aligned}$$

$$\text{Thus, } H(k) = \sum_{j=0}^{k} U(j)V(k-j) \tag{3}$$

over all nonnegative integers k. This sum of U and V values has been called a discrete convolution and this combination of series has been called a Cauchy product. Variations of this rule will be the source of all of our recurrence formulas.

The Leibniz rule for the kth-order derivative of a product introduces binomial coefficients in each term and is less efficient than the Cauchy product. An alternative approach to this paper would apply the Leibniz rule to arrays of derivative values at a, which are larger by a factorial factor from the U and V coefficient values. Computing with such large derivative values has been seen to produce more cumulative roundoff error than using Taylor coefficient values.

The transform of a **quotient** follows from the product. If $h(x) = u(x)/v(x)$, then $u(x) = h(x)\, v(x)$, so

$$\begin{aligned} U(k) &= \sum_{j=0}^{k} H(j)V(k-j) \\ &= H(k)V(0) + \sum_{j=0}^{k-1} H(j)V(k-j). \end{aligned}$$

$$\text{Thus, } H(k) = \frac{1}{V(0)}\left(U(k) - \sum_{j=0}^{k-1} H(j)V(k-j)\right). \tag{4}$$

This recurrence relation, for nonnegative integers k, can be used to divide by any $v\,(x)$ where $V(0) = v(a) \neq 0$. It may be generalized to deal with analytic functions such as $\frac{\sin(x)}{x}$ or $\frac{e^x-1}{x}$ about $a = 0$. If $U(0) = V(0) = 0$, then $h(x) = \frac{u(x)/(x-a)}{v(x)/(x-a)}$ is a quotient of two power series with the coefficient values simply shifted to the left. Unfortunately, if coefficients in U and V are only known up to order m, then coefficients in H are only produced up to order $m-1$. Subsequent shifts could be implemented if further terms vanish in both numerator and denominator.

2.2. **Composition & the DE theorem.** Consider a composition $h(x) = f(u(x))$ where $f : \mathbb{R} \to \mathbb{R}$ is one of the standard functions found on a typical scientific calculator, including exponential and trigonometric functions and their inverses. Then

$$\begin{aligned} h'(x) &= f'(u(x))u'(x) \text{ or} \\ h'(x) &= v(x)\, u'(x). \end{aligned}$$

For a simple $v(x) = f'(u(x))$, we use U and generate V along with H. For example, if $h(x) = \exp(u(x))$, then $v = h$ and $V = H$. In other cases, we get one, or a system of two, differential equations of the above form although the roles of u, v, and h may be switched. Examples of this appear in Section 2.3. The point is that we will be using the specific form $h'(x) = v(x)\,u'(x)$ resulting from a specific f. We avoid the use of a higher-order chain rule for arbitrary (symbolic) function f, the Faà di Bruno formula, which involves significant combinatorial complications.

For $h'(x) = v(x)\,u'(x)$, consider a Cauchy product involving derivatives of the corresponding series

$$\sum_{k=1}^{\infty} H(k)\,k(x-a)^{k-1} = \left(\sum_{i=0}^{\infty} V(i)(x-a)^i\right)\left(\sum_{j=1}^{\infty} U(j)\,j(x-a)^{j-1}\right).$$

Collecting the coefficient of $(x-a)^{k-1}$ on each side yields

$$k\,H(k) = \sum_{i+j=k} jU(j)V(i) = \sum_{j=1}^{k} jU(j)V(k-j).$$

For our recurrence rules, we pull out the $j = k$ term and divide by k:

$$H(k) = V(0)U(k) + \frac{1}{k}\sum_{j=1}^{k-1} jU(j)V(k-j). \tag{5}$$

For repeated application to the differential equation (DE) form $h'(x) = v(x)\,u'(x)$ and to facilitate a multivariate generalization, we encapsulate (5), without the isolated term, as a computational function `ddot` and state the resulting principle as a theorem.

Theorem 2.1 (DE Theorem)**.** *If $h'(x) = v(x)u'(x)$ (on a neighborhood of a where all these functions are analytic) and $k \in \mathbb{N}$, then $H(k) = V(0)U(k) + \texttt{ddot}(V, U, k)$ where*

$$\texttt{ddot}(V, U, k) = \frac{1}{k}\sum_{j=1}^{k-1} jU(j)V(k-j).$$

Notice that $\texttt{ddot}(V, U, k)$ uses V and U coefficients only from 1 to $k-1$, which is important in order to avoid a self-referential definition of $H(k)$, even if H plays the role of V and/or U. Recall that $jU(j)$ comes from the derivative of u, so this "derivative dot product" is a kind of Cauchy product of v and the derivative of u. It will be convenient to use the principle with a scalar multiple:

$$\begin{aligned} &\text{if } h'(x) = \alpha\, v(x)\, u'(x) \\ &\text{then } H(k) = \alpha\,\left(V(0)U(k) + \texttt{ddot}(V, U, k)\right). \end{aligned} \tag{6}$$

2.3. **Standard functions.** Using the DE Theorem, each standard (calculator button) function results in a recurrence relation for the series coefficients when the function is composed with one whose series coefficients are already known.

If $h(x) = \exp(u(x))$, $H(0) = \exp(U(0))$ and $h'(x) = h(x)\,u'(x)$, so for $k \in \mathbb{N}$,

$$H(k) = H(0)U(k) + \texttt{ddot}(H, U, k). \tag{7}$$

If $h(x) = \ln(u(x))$, $H(0) = \ln(U(0))$. Since $h'(x) = u'(x)/u(x)$, we write $u'(x) = u(x)\,h'(x)$. By the DE Theorem $U(k) = U(0)H(k) + \texttt{ddot}(U, H, k)$ for

$k \in \mathbb{N}$, and we conclude

$$H(k) = (U(k) - \texttt{ddot}(U, H, k))\,/U(0).$$

Trigonometric functions and inverse trigonometric functions produce coupled recurrence relations from the DE Theorem. From now on, we will assume that the zero term is initialized as the function value and that the recurrence relation applies for $k \in \mathbb{N}$.

If $s(x) = \sin(u(x))$ and $c(x) = \cos(u(x))$, $s'(x) = c(x)\,u'(x)$ and $c'(x) = -s(x)\,u'(x)$, so

$$S(k) = C(0)U(k) + \texttt{ddot}(C, U, k) \text{ and}$$
$$C(k) = -S(0)U(k) - \texttt{ddot}(S, U, k).$$

If $t(x) = \tan(u(x))$, let $v(x) = \sec^2(u(x)) = 1 + \tan^2(u(x))$. Then $t'(x) = v(x)\,u'(x)$ and $v'(x) = 2t(x)\,t'(x)$, so

$$T(k) = V(0)U(k) + \texttt{ddot}(V, U, k) \text{ and}$$
$$V(k) = 2\,(T(0)T(k) + \texttt{ddot}(T, T, k))\,.$$

If $h(x) = \arctan(u(x))$, let $v(x) = 1 + (u(x))^2$. Then $h'(x) = u'(x)/v(x)$, so $u'(x) = v(x)\,h'(x)$, and $U(k) = V(0)H(k) + \texttt{ddot}(V, H, k)$. Solving for $H(k)$ and using $v'(x) = 2u(x)\,u'(x)$, we conclude

$$H(k) = (U(k) - \texttt{ddot}(V, H, k))/V(0) \text{ and}$$
$$V(k) = 2\,(U(0)U(k) + \texttt{ddot}(U, U, k))\,.$$

If $h(x) = \arcsin(u(x))$, $u(x) = \sin(h(x))$. Let $v(x) = \cos(h(x))$. Then $u'(x) = v(x)\,h'(x)$, so $U(k) = V(0)H(k) + \texttt{ddot}(V, H, k)$. Solving for $H(k)$ and using $v'(x) = -u(x)\,h'(x)$,

$$H(k) = (U(k) - \texttt{ddot}(V, H, k))/V(0) \text{ and}$$
$$V(k) = -U(0)H(k) - \texttt{ddot}(U, H, k).$$

Surprisingly, $h(x) = \sec(u(x))$ seems to require three such operations. We can compute S and C for $c(x) = \cos(u(x))$ and apply (4) to $h(x) = 1/c(x)$. Likewise, many functions, can be built from operations above. Hyperbolic functions can be built from exp operations but sometimes it is more efficient to derive the coupled recurrence relations from the DE Theorem by mimicking the above derivations.

2.4. **Exponents.** For the most general exponentiation $h(x) = u(x)^{v(x)}$ with positive $u(x)$, one would use $h(x) = \exp(v(x)\ln(u(x)))$. For each step we have recurrences using three generalized Cauchy products:

$$\begin{aligned} h_1(x) &= \ln u(x) & H_1(k) &= (U(k) - \texttt{ddot}(U, H_1, k))\,/U(0) \\ h_2(x) &= v(x)h_1(x) & H_2(k) &= \textstyle\sum_{j=0}^{k} H_1(j)V(k-j) \\ h(x) &= \exp(h_2(x)) & H(k) &= H(0)H_2(k) + \texttt{ddot}(H, H_2, k) \end{aligned}$$

If U and V are completely known, operations in $\exp(V * \ln(U))$ could be overloaded to perform three loops, one on each line the above recurrences; once computed, intermediate results can be discarded. However, for a differential equation like $u' = u^v$, once through the sequence of three recurrences would return the next series coefficient and the process would be repeated saving intermediate results.

The above uses three generalized Cauchy products, but many important special exponent cases use only one such product:

- $h(x) = u(x)^2$ is $h(x) = u(x)\,u(x)$ or use $h'(x) = 2u(x)\,u'(x)$.

- $h(x) = \sqrt{u(x)}$ can use $u(x) = h(x)\,h(x)$ or $u'(x) = 2h(x)\,h'(x)$.
- $h(x) = u(x)^{-1}$ can use $h(x) = 1/u(x)$ or, equivalently, $h(x)\,u(x) = 1$.
- $h(x) = r^{u(x)}$ is $h(x) = e^{(\ln r)u(x)}$ or use $h'(x) = (\ln r)h(x)\,u'(x)$.

For $h(x) = u(x)^r$, the important case of constant exponent, we use only two generalized Cauchy products. Since $h'(x) = r\,u(x)^{r-1}u'(x)$, we have $u(x)\,h'(x) = r\,h(x)\,u'(x)$. Using the DE Theorem (for $F(k)$ where $f'(x) =$ both sides of above), we get $U(0)H(k) + \texttt{ddot}(U, H, k) = r\,[H(0)U(k) + \texttt{ddot}(H, U, k)]$. Thus

$$H(k) = \{r\,[H(0)U(k) + \texttt{ddot}(H, U, k)] - \texttt{ddot}(U, H, k)\}\,/U(0).$$

3. **Multivariate recurrences.** Every $h : \mathbb{R}^n \to \mathbb{R}$ analytic in a neighborhood of $\mathbf{a}$, transforms to $H : \mathbb{N}_0^n \to \mathbb{R}$, the Taylor coefficient function, such that

$$\begin{aligned} h(\mathbf{x}) &= \textstyle\sum_{\mathbf{k}} H(\mathbf{k})(\mathbf{x} - \mathbf{a})^{\mathbf{k}} \\ &= \textstyle\sum_{\mathbf{k}} H(\mathbf{k})(x_1 - a_1)^{k_1}(x_2 - a_2)^{k_2} \cdots (x_n - a_n)^{k_n} \end{aligned}$$

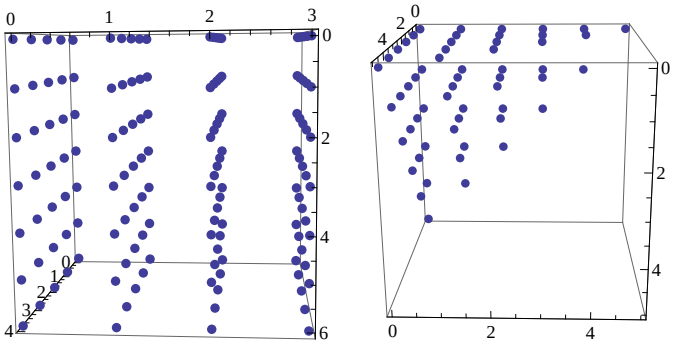
over all *multi-indices* $\mathbf{k} = (k_1, k_2, \ldots, k_n)$ of nonnegative integers. We rarely use the Taylor coefficient formula

$$H(\mathbf{k}) = \frac{1}{k_1!k_2!\cdots k_n!}\left(\frac{\partial}{\partial x_1}\right)^{k_1}\left(\frac{\partial}{\partial x_2}\right)^{k_2}\cdots\left(\frac{\partial}{\partial x_n}\right)^{k_n} h\,(a)\,.$$

As in univariate, transforms of elementary functions can be built from operations on the most basic cases, where $\mathbf{e}_i$ denotes the multi-index with 1 in the ith coordinate and zeros in all others:

- If $h(\mathbf{x}) = \alpha \in \mathbb{R}$, then $H(\mathbf{0}) = \alpha$ and $H(\mathbf{k}) = 0$ otherwise.
- If $h_i(\mathbf{x}) = x_i$, then $H_i(\mathbf{0}) = a_i$, $H_i(\mathbf{e}_i) = 1$ and $H_i(\mathbf{k}) = 0$ otherwise.

Each transform can be represented in a computer language as an object holding an array of coefficient values over some finite restriction of the multi-index domain, usually a box or corner as shown in Figure 1, using $|\mathbf{j}| = j_1 + j_2 + j_3$. For a fixed



(a) $H(\mathbf{j})$ for $\mathbf{j} \le (6, 3, 4)$ (b) $H(\mathbf{j})$ for $|\mathbf{j}| \le 5$

FIGURE 1. Three variable multi-indices for Taylor coefficients.

number of variables, many languages allow working with a multi-dimensional box array, although these can grow huge very quickly. Figure 1(a), with multi-indices $\le (6, 3, 4)$ coordinatewise, represents Taylor coefficients that correspond to at most 6 derivatives in x_1, 3 derivatives in x_2, and 4 derivatives in x_3. Computing up to a fixed order, as in Figure 1(b), is probably best implemented by a linear array, where

each linear index corresponds to a multi-index. This is the biggest complication in multivariate work. We will focus on the conceptually simple task of finding recurrence relations to compute $H(\mathbf{k})$, assuming $H(\mathbf{j})$ have already been computed for all multi-indices $\mathbf{j} \le \mathbf{k}$ coordinatewise with $\mathbf{j} \ne \mathbf{k}$. Processing array entries in such an order is easy, but linear indexing makes it more difficult to access needed subsets of previous values.

3.1. **Arithmetic operations.** The multivariate Taylor coefficient transform is also linear: if $h(\mathbf{x}) = \alpha\, u(\mathbf{x}) + \beta\, v(\mathbf{x})$, then $H = \alpha U + \beta V$, as in matrix algebra.

For a product $h(\mathbf{x}) = u(\mathbf{x})\, v(\mathbf{x})$,

$$\begin{aligned} h(\mathbf{x}) &= \left(\textstyle\sum_{\mathbf{k}} U(\mathbf{k})(\mathbf{x}-\mathbf{a})^{\mathbf{k}}\right)\left(\textstyle\sum_{\mathbf{k}} V(\mathbf{k})(\mathbf{x}-\mathbf{a})^{\mathbf{k}}\right) \\ &= \textstyle\sum_{\mathbf{k}} \left(\displaystyle\sum_{\mathbf{j}\le\mathbf{k}} U(\mathbf{j})V(\mathbf{k}-\mathbf{j})\right)(\mathbf{x}-\mathbf{a})^{\mathbf{k}} \end{aligned}$$

so

$$H(\mathbf{k}) = \sum_{\mathbf{j}\le\mathbf{k}} U(\mathbf{j})V(\mathbf{k}-\mathbf{j}) \tag{8}$$

summing over all multi-indices $\mathbf{j} \le \mathbf{k}$ coordinatewise. This is a multivariate generalization of the Cauchy product. If $\mathbf{k} = (6,3,4)$, the sum is over the whole box of entries, shown in Figure 1(a), with U and V arrays processed in opposite order. For $\mathbf{k} = (2,1,2)$ (in Figure 1(a) or (b)), the sum would be over the $3\times 2\times 3$ sub-box of entries.

For a quotient $h(\mathbf{x}) = u(\mathbf{x})/v(\mathbf{x})$, write $u(\mathbf{x}) = h(\mathbf{x})\, v(\mathbf{x})$, so

$$\begin{aligned} U(\mathbf{k}) &= \sum_{\mathbf{j}\le\mathbf{k}} H(\mathbf{j})V(\mathbf{k}-\mathbf{j}) \\ &= H(\mathbf{k})V(\mathbf{0}) + \sum_{\mathbf{j}\lneq\mathbf{k}} H(\mathbf{j})V(\mathbf{k}-\mathbf{j}) \end{aligned}$$

summing over all multi-indices $\mathbf{j} \le \mathbf{k}$ coordinatewise with $\mathbf{j} \ne \mathbf{k}$. Thus

$$H(\mathbf{k}) = \frac{1}{V(\mathbf{0})}\left(U(\mathbf{k}) - \sum_{\mathbf{j}\lneq\mathbf{k}} H(\mathbf{j})V(\mathbf{k}-\mathbf{j})\right).$$

3.2. **Multivariate DE theorem.** For the composition $h(\mathbf{x}) = f(u(\mathbf{x}))$ where $f : \mathbb{R} \to \mathbb{R}$ is one of the standard (calculator button) functions that generate the set of elementary functions,

$$\begin{aligned} \frac{\partial h}{\partial x_i}(\mathbf{x}) &= f'(u(\mathbf{x}))\frac{\partial u}{\partial x_i}(\mathbf{x}) \text{ or} \\ \frac{\partial h}{\partial x_i}(\mathbf{x}) &= v(\mathbf{x})\,\frac{\partial u}{\partial x_i}(\mathbf{x}), \text{ for each } i, \end{aligned}$$

which plays the same role as $h'(x) = v(x)\, u'(x)$ in univariate derivations.

Theorem 3.1 (Multivariate DE Theorem)**.** *If* $\frac{\partial h}{\partial x_i}(\mathbf{x}) = v(\mathbf{x})\frac{\partial u}{\partial x_i}(\mathbf{x})$ *for each* i *(on a neighborhood of* $\mathbf{a}$ *where all these functions are analytic) and* $\mathbf{k} \ne \mathbf{0}$*, choose a coordinate* p *such that* $k_p = \min\{k_i : k_i \ne 0\}$*. Then* $H(\mathbf{k}) = V(\mathbf{0})U(\mathbf{k}) + \texttt{ddot}(V,U,\mathbf{k})$ *where*

$$\texttt{ddot}(V,U,\mathbf{k}) = \frac{1}{k_p}\sum_{\mathbf{e}_p\le\mathbf{j}\lneq\mathbf{k}} j_p U(\mathbf{j})V(\mathbf{k}-\mathbf{j})$$

summing over all multi-indices with coordinates $j_p = 1, 2, \ldots, k_p$ *and all other* $j_i = 0, 1, \ldots, k_i$ *except that* $\mathbf{j} = \mathbf{k}$ *is omitted.*

Proof. For $\mathbf{k} \neq \mathbf{0}$, we may choose any coordinate p such that $k_p \neq 0$. We suggest $k_p = \min\{k_i : k_i \neq 0\}$, in order to minimize the number of terms in the `ddot` summation that results. By hypothesis $\frac{\partial h}{\partial x_p}(\mathbf{x}) = v(\mathbf{x})\frac{\partial u}{\partial x_p}(\mathbf{x})$. Thus

$$\frac{\partial}{\partial x_p}\left(\sum_{\mathbf{k}} H(\mathbf{k})(\mathbf{x}-\mathbf{a})^{\mathbf{k}}\right) = \left(\sum_{\mathbf{i}} V(\mathbf{i})(\mathbf{x}-\mathbf{a})^{\mathbf{i}}\right)\frac{\partial}{\partial x_p}\left(\sum_{\mathbf{j}} U(\mathbf{j})(\mathbf{x}-\mathbf{a})^{\mathbf{j}}\right) \text{ and}$$
$$\sum_{\mathbf{k}} H(\mathbf{k})\, k_p (\mathbf{x}-\mathbf{a})^{\mathbf{k}-\mathbf{e}_p} = \left(\sum_{\mathbf{i}} V(\mathbf{i})(\mathbf{x}-\mathbf{a})^{\mathbf{i}}\right)\left(\sum_{\mathbf{j}} U(\mathbf{j})\, j_p (\mathbf{x}-\mathbf{a})^{\mathbf{j}-\mathbf{e}_p}\right).$$

Collecting the coefficient of $(\mathbf{x}-\mathbf{a})^{\mathbf{k}-\mathbf{e}_p}$ on each side yields

$$\begin{aligned} k_p\, H(\mathbf{k}) &= \sum_{\mathbf{i}+\mathbf{j}=\mathbf{k}} j_p U(\mathbf{j}) V(\mathbf{i}) \text{ so} \\ H(\mathbf{k}) &= \frac{1}{k_p}\sum_{\mathbf{e}_p \leq \mathbf{j} \leq \mathbf{k}} j_p U(\mathbf{j}) V(\mathbf{k}-\mathbf{j}) \\ &= V(\mathbf{0})U(\mathbf{k}) + \frac{1}{k_p}\sum_{\mathbf{e}_p \leq \mathbf{j} \lneq \mathbf{k}} j_p U(\mathbf{j}) V(\mathbf{k}-\mathbf{j}) \\ &= V(\mathbf{0})U(\mathbf{k}) + \texttt{ddot}(V, U, \mathbf{k}). \end{aligned}$$

□

Geometrically, `ddot` can be envisioned by considering the sub-box of U for $\mathbf{j} \leq \mathbf{k}$, sliced by fixed p-coordinate. Drop the first such face, then other slices are dotted with the reverse of the opposite slice of V, results are multiplied by j_p and summed. For example, if $n = 2$ and $\mathbf{k} = (6, 3)$, the entries needed in V and U are shown in Figure 2. By choosing $k_p = k_2 = 3$, we drop the largest face of this box (which could

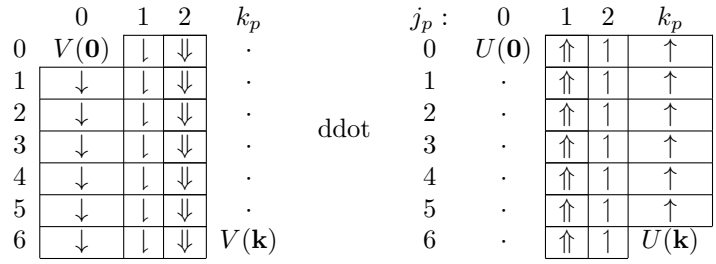


FIGURE 2. Visualizing `ddot`$(V, U, \mathbf{k})$.

extend into even more dimensions for larger n). Entries are multiplied and summed as indicated by the arrows. When each slice is finished the result is multiplied by j_p and the final total is divided by k_p.

The `ddot` operation and Cauchy product can be made very efficient by storing a large global reference list (of matrices of indices) that depends only on the number of variables n and the highest order m of desired Taylor coefficients, as in Figure 1(b). A linear index can progress through entries corresponding to each multi-index $\mathbf{k}$. For each linear index, the global reference can store linear indices corresponding to the sub-box of all $\mathbf{j} \leq \mathbf{k}$, divided into slices; making a 2D matrix of indices for each $\mathbf{k}$, regardless of n. Then `ddot` simply runs through the matrix of indices, as

diagrammed in Figure 2. Assuming the linear ordering of the n-dimensional multi-indices is increasing either lexically and/or in $|\mathbf{k}|$ order, values on the sub-box will be computed before a value for $\mathbf{k}$. This process is implemented in MATLAB in [1], where the global reference of sub-boxes was much faster to compute by recurrence, than with direct search.

3.3. **Standard functions.** After details are encapsulated in `ddot`, the multivariate DE theorem looks exactly like the univariate version in (6);

$$\begin{aligned} &\text{if } \frac{\partial h}{\partial x_i}(\mathbf{x}) = \alpha\, v(\mathbf{x}) \frac{\partial u}{\partial x_i}(\mathbf{x}) \text{ for all } i, \\ &\text{then } H(\mathbf{k}) = \alpha\, (V(\mathbf{0})U(\mathbf{k}) + \texttt{ddot}(V, U, \mathbf{k}))\,. \end{aligned} \tag{9}$$

With this understanding, all of the standard (calculator button) functions have the same transform rules when applied to multivariate functions.

Consider $h(\mathbf{x}) = \exp(u(\mathbf{x}))$ where $u : \mathbb{R}^n \to \mathbb{R}$ and the multivariate transform U is known. In order to see how this starts, consider $u(x, y, z) = xyz$ about (a, b, c). The basic arrays X, Y, and Z would have just two nonzero entries, like $Y((0, 0, 0)) = b$, $Y((0, 1, 0)) = 1$. The Cauchy product rule (8) is used twice to find $U = X * Y * Z$. For any u and $h(\mathbf{x}) = \exp(u(\mathbf{x}))$, we have $\frac{\partial h}{\partial x_i}(\mathbf{x}) = h(\mathbf{x}) \frac{\partial u}{\partial x_i}(\mathbf{x})$ for all i, playing the same role as $h'(x) = h(x)u'(x)$ in the univariate case. Then $H(\mathbf{0}) = \exp(U(\mathbf{0}))$ and all other $H(\mathbf{k})$ values are given by (9), specifically

$$H(\mathbf{k}) = H(\mathbf{0})U(\mathbf{k}) + \texttt{ddot}(H, U, \mathbf{k}). \tag{10}$$

Just as the only difference between (7) and (10) is that the index arguments are multivariate, all of the `ddot` rules of Sections 2.3 and 2.4 apply to multivariate arguments as well, and will not be repeated here. Of course, the implementation of indexing, arrays, and `ddot` are more complicated but the standard function rule is just the same. There is a new motivation in the multivariate case: a `ddot` rule (9) can be more efficient than a Cauchy product rule (8) since `ddot` omits an entire face from of each sum of products, as described at the end of the last section. While $h(x) = u(x)^2$ could use either $h(x) = u(x)\, u(x)$ or $h'(x) = 2u(x)\, u'(x)$, the latter would be preferred for multivariate work. The square root function would be done in a similar way.

If $h(\mathbf{x}) = u(\mathbf{x})^2$, then $\frac{\partial h}{\partial x_i}(\mathbf{x}) = 2\, u(\mathbf{x})\, \frac{\partial u}{\partial x_i}(\mathbf{x})$, so

$$H(\mathbf{k}) = 2\, (U(\mathbf{0})U(\mathbf{k}) + \texttt{ddot}(U, U, \mathbf{k}))\,.$$

If $h(x) = \sqrt{u(x)}$, then $\frac{\partial h}{\partial x_i}(\mathbf{x}) = \frac{1}{2h(\mathbf{x})} \frac{\partial u}{\partial x_i}(\mathbf{x})$, or $\frac{\partial u}{\partial x_i}(\mathbf{x}) = 2\, h(\mathbf{x})\, \frac{\partial h}{\partial x_i}(\mathbf{x})$. By the multivariate DE theorem, $U(\mathbf{k}) = 2\, (H(\mathbf{0})H(\mathbf{k}) + \texttt{ddot}(H, H, \mathbf{k}))$, so

$$H(\mathbf{k}) = (U(\mathbf{k})/2 - \texttt{ddot}(H, H, \mathbf{k}))/H(\mathbf{0}).$$

4. **Application to an ODE.** To see how these rules are used to compute a high-order solution to an ordinary differential equation with initial condition, we return to the univariate setting and consider the example problem $y' = -xy - \sin(y)$, $y(x_0) = y_0$. Initial values specify the transform (or Taylor coefficients) for $X = (x_0, 1, 0, 0, \ldots)$ and the first two terms $Y(0) = y_0$, $Y(1) = -x_0 y_0 - \sin(y_0)$. To produce further terms of Y, we break the right-hand-side of the differential equation into steps with one variable name for each product (division) or standard function and any trigonometric cofunctions as in Section 2.3. Let $p = xy$, $s = \sin(y)$, $c = \cos(y)$, and $y' = -p - s$, which is called a code list in AD. The transform of each of these variables is initialized: $P(0) = x_0 y_0$, $S(0) = \sin(y_0)$, $C(0) = \cos(y_0)$. The

differential equation transforms to $(k+1)Y(k+1) = -P(k) - S(k)$, where $P(k)$ and $S(k)$ are computed by Section 2.3 rules. In this case, $P(k) = \sum_{j=0}^{k} X(j)Y(k-j) = x_0 Y(k) + Y(k-1)$. For $k \geq 1$, compute arbitrarily many terms of Y by iterating through

$$\begin{aligned} P(k) &= x_0 Y(k) + Y(k-1), \\ S(k) &= C(0)Y(k) + \texttt{ddot}(C, Y, k), \\ C(k) &= -S(0)Y(k) - \texttt{ddot}(S, Y, k), \\ Y(k+1) &= (-P(k) - S(k))/(k+1), \end{aligned}$$

which could be simplified by substitution for $P(k)$ in the last step. Such a sequence of recurrence relations is the Differential Transform Method solution. The solution function $y(x) \approx \sum_{k=0}^{m} Y(k)(x - x_0)^k$ gives y almost to the radius of convergence for large m. Calculation for $y(-1) = 2$ shows the 50th order Taylor polynomial visually agreeing with the ode45 MATLAB solution for a radius of 1.5, and the 10th order for over radius 1.0. Numerical or a-priori error estimates could detect such a radius and use it as a large step to a new point, where the recurrence is restarted and another series is found. This algorithmic version of a high-order Taylor series method does not require symbolic differentiation.

REFERENCES

[1] B. Altman, "Higher-Order Automatic Differentiation of Multivariate Functions in MATLAB," Undergraduate Honors Thesis, Davidson College, 2010.

[2] F. Dangello and M. Seyfried, "Introductory Real Analysis," Houghton Mifflin, 2000, Section 7.4.

[3] W. Dunham, "Euler: The Master of Us All," MAA, 1999, Chapter 3.

[4] A. Griewank and A. Walther, "Evaluating Derivatives," 2nd edition, SIAM, 2008, Section 13.2.

[5] M-J. Jang, C-L. Chen and Y-C. Liu, *Two-dimensional differential transform for partial differential equations*, Appl. Math. Computation, **121** (2001), 261–270.

[6] R.E. Moore, "Methods and Applications of Interval Analysis," SIAM, 1979, Section 3.4.

[7] R.D. Neidinger, *Computing multivariable Taylor series to arbitrary order*,, APL Quote Quad, **25** (1995), 134–144.

[8] R.D. Neidinger, *Automatic differentiation and MATLAB object-oriented programming*, SIAM Review, **52** (2010), 545-563.

[9] G.E. Parker and J.S. Sochacki, *Implementing the Picard iteration*, Neural, Parallel and Scientific Computation, **4** (1996), 97-112.

[10] L.B. Rall, *Early automatic differentiation: the Ch'in-Horneralgorithm*, Reliable Computing, **13** (2007), 303-308.

[11] J. Waldvogel, *Der Tayloralgorithmus*, J. Applied Math. and Physics (ZAMP), **35** (1984), 780-789.

Received July 2012; revised April 2013.

E-mail address: rineidinger@davidson.edu