Evaluating higher derivative tensors by forward propagation of univariate Taylor series

Andreas Griewank*
Jean Utke§
Andrea Walther*

23rd June 1998

Abstract

This article considers the problem of evaluating all pure and mixed partial derivatives of some vector function defined by an evaluation procedure. The natural approach to evaluate derivative tensors might appear to be their recursive calculation in the usual forward mode of Computational Differentiation. However, with the approach presented in this article, much simpler data access patterns and similar or lower computational counts can be achieved through propagating a family of univariate Taylor series of a suitable degree. It is applicable for arbitrary orders of derivatives. Also it is possible to calculate derivatives only in some directions instead of the full derivative tensor. Explicit formulas for all tensor entries as well as estimates for the corresponding computational complexities are given.

1991 Mathematics Subject Classification: Primary 65D05, 65Y20, 68Q40
Keywords: Higher Order Derivatives, Computational Differentiation

1 Introduction

Many applications in scientific computing require second- and higher-order derivatives. Therefore, this article deals with the problem of calculating derivative tensors of some vector function

$$\boldsymbol{y} \quad = \quad \boldsymbol{f}(\boldsymbol{x}) \qquad \text{with } \boldsymbol{f} : \mathcal{D} \subset \mathbb{R}^{\bar{n}} \mapsto \mathbb{R}^{m},$$

that is the composition of (at least locally) smooth elementary functions. Assume that $\boldsymbol{f}$ is given by an evaluation procedure in C or some other programming language. Then $\boldsymbol{f}$ can be differentiated automatically [Gri89]. The Jacobian matrix of first partial derivatives can be computed by the forward or reverse mode of the chain-rule based technique known commonly as Computational Differentiation (CD). CD also yields second derivatives that are needed in optimization [Gri94] and even higher derivatives that are called for in numerical bifurcation, beam dynamics [Ber89] and other nonlinear calculations.

*Institute of Scientific Computing, Technical University Dresden, D-01062 Dresden, Germany
§Mathematics and Computer Science Devision, Argonne National Laboratory, Argonne, IL 60439, USA

Even though the reverse mode of CD may be more efficient when the number of dependent variables is small compared to the number of independents [Gri89], only the forward mode will be considered here. The mechanics of this direct application of the chain rule are completely independent of the number of dependent variables so that it is possible to restrict the analysis to a scalar-valued function

$$y \;=\; f(\boldsymbol{x}) \quad \text{with} \quad f : \mathcal{D} \subset \mathbb{R}^{\bar{n}} \mapsto \mathbb{R}.$$

In other words, formulas for one component $f(\boldsymbol{x})$ of the original vector function $\boldsymbol{f}(\boldsymbol{x})$ are derived. This greatly simplifies the notation, and the full tensors can then easily be obtained by an outer loop over the component index.

The natural approach to evaluate derivative tensors seems to be their recursive calculation using the usual forward mode of CD. This technique has been implemented by Berz [Ber91], Neidinger [Nei92], and others. The only complication here is the need to utilize the symmetry in the higher derivative tensors, which leads to fairly complex addressing schemes. As has been mentioned in [Gri91] and [BCG93], much simpler data access patterns and similar or lower computational counts can be achieved through propagating a family of univariate Taylor series of an arbitrary degree. At the end, their values are interpolated to yield the tensor coefficients.

The paper is organized as follows. Section 2 introduces the notations that are used and makes some general observations. In Section 3 the complexity of storing and propagating multivariate and univariate Taylor polynomials is examined and the advantages of the univariate Taylor series are shown. Section 4 derives formulas for the calculation of all mixed partial derivatives up to degree d from a family of univariate Taylor polynomials of degree d. Some run-time results are presented in Section 5. Finally, Section 6 outlines some conclusions.

2 Notations and basic observations

In many applications, one does not require full derivative tensors but only the derivatives in $n \le \bar{n}$ directions $\boldsymbol{s}_i \in \mathbb{R}^{\bar{n}}$. Therefore suppose a collection of $n \le \bar{n}$ directions $\boldsymbol{s}_i \in \mathbb{R}^{\bar{n}}$ is given that form a matrix

$$\boldsymbol{S} \;=\; [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n] \;\in\; \mathbb{R}^{\bar{n}\times n}.$$

One possible choice is $\boldsymbol{S} = \boldsymbol{I}$ with $n = \bar{n}$. Here, $\boldsymbol{I}$ denotes the identity matrix $\boldsymbol{I}$ in $\mathbb{R}^{n\times n}$. Of particular interest is the case $n = 1$ where only derivatives in one direction are calculated.

With a coefficient vector $\boldsymbol{z} \in \mathbb{R}^n$, one may wish to calculate the derivative tensors

$$\begin{aligned}
\left.\frac{\partial}{\partial \boldsymbol{z}} f(\boldsymbol{x} + \boldsymbol{S}\boldsymbol{z})\right|_{\boldsymbol{z}=0} &= \left(f'(\boldsymbol{x})\boldsymbol{s}_j\right)_{j=1,\ldots,n} \in \mathbb{R}^n \\
\left.\frac{\partial^2}{\partial \boldsymbol{z}^2} f(\boldsymbol{x} + \boldsymbol{S}\boldsymbol{z})\right|_{\boldsymbol{z}=0} &= \left(f''(\boldsymbol{x})\, \boldsymbol{s}_i\, \boldsymbol{s}_j\right)_{i=1,\ldots,n}^{j=1,\ldots,n} \in \mathbb{R}^{n\times n}
\end{aligned}$$

and so on. The last equation is already an abuse of the usual matrix-vector notation. Here, the abbreviation

$$\nabla_{\boldsymbol{S}}^k f(\boldsymbol{x}) \in \mathbb{R}^{n^k}$$

will be used in order to denote the k-th derivative tensor of $f(\boldsymbol{x} + \boldsymbol{S}\boldsymbol{z})$ with respect to $\boldsymbol{z}$ at $\boldsymbol{z} = 0$.

The use of the *seed matrix* $\boldsymbol{S}$ allows the restriction of the consideration to a subspace spanned by the columns $\boldsymbol{s}_i$ along which the properties of f might be particularly interesting. This situation arises for example in optimization and bifurcation theory, where the range of $\boldsymbol{S}$ might be the tangent space of the active constraints or the nullspace of a degenerate Jacobian. Especially, when $n \ll \bar{n}$ it is obviously preferable to calculate the restricted tensors $\nabla_{\boldsymbol{S}}^k f(\boldsymbol{x})$ directly rather than first evaluating the full tensors

$$\nabla^k f(\boldsymbol{x}) \;\equiv\; \nabla_{\boldsymbol{I}}^k f(\boldsymbol{x}) \;\in\; \mathbb{R}^{\bar{n}^k}$$

and then contracting them by multiplications with the directions $\boldsymbol{s}_i$.

One way to evaluate the desired tensors $\nabla_{\boldsymbol{S}}^k f(\boldsymbol{x})$ is to recursively calculate for all intermediate quantities w the corresponding restricted tensors $\nabla_{\boldsymbol{S}} w, \nabla_{\boldsymbol{S}}^2 w, \ldots$ etc. The process would be started with the initialization $\nabla_{\boldsymbol{S}} \boldsymbol{x} = \boldsymbol{S}$ and with $\nabla_{\boldsymbol{S}}^k \boldsymbol{x} = 0$ for $k > 1$. For all subsequent intermediates, the derivative tensors are defined by the usual differentiation rules. For example, in the case $w = u * v$ there are the well known differentiation rules

$$\nabla_{\boldsymbol{S}} w \;=\; u\, \nabla_{\boldsymbol{S}} v + v\, \nabla_{\boldsymbol{S}} u$$

and

$$\nabla_{\boldsymbol{S}}^2 w \;=\; u\, \nabla_{\boldsymbol{S}}^2 v + \nabla_{\boldsymbol{S}} u (\nabla_{\boldsymbol{S}} v)^T + \nabla_{\boldsymbol{S}} v (\nabla_{\boldsymbol{S}} u)^T + v\, \nabla_{\boldsymbol{S}}^2 u.$$

For the third derivative tensor, the matrix-vector notation is no longer sufficient, but the following observation are generally applicable.

To calculate $\nabla_{\boldsymbol{S}}^k w$, each element of $\nabla_{\boldsymbol{S}}^j u$ with $0 \leq j \leq k$ has to be considered and then multiplied with all elements of $\nabla_{\boldsymbol{S}}^{k-j} v$. The result is then multiplied by a binomial coefficient given by Leibniz' theorem and finally incremented to the appropriate element of $\nabla_{\boldsymbol{S}}^k w$.

The next section studies the complexity of storing and propagating multivariate and univariate Taylor polynomials. Taylor coefficients are used since they are somewhat better scaled than the corresponding derivatives and satisfy slightly simpler recurrences.

3 The complexity of propagating univariate Taylor series

If the k-th tensor was stored and manipulated as a full n^k array, the corresponding loops could be quite simple to code, but the memory requirement and operations count would be unnecessary high. In the case $k = 2$, ignoring the symmetry of Hessians would almost double the storage per intermediate (from $\frac{1}{2}(n+1)n$ to n^2) and increase the operations count for a multiplication by about fifty percent. This price may be worth paying in return for the resulting sequential or at least constant stride data access. However, by standard combinatorial arguments

$$\nabla_{\boldsymbol{S}}^k f(\boldsymbol{x}) \;\in \mathbb{R}^{n^k}$$

has exactly

$$\binom{n+k-1}{k} \;=\; \frac{n \cdot (n+1) \cdots (n+k-1)}{1 \cdot 2 \cdots k} \;\approx\; \frac{n^k}{k!} \tag{1}$$

distinct elements. Hence, the symmetry reduces the number of distinct elements in $\nabla_S^k w$ almost by the factor $k!$. Therefore in the case $k = 3$ the number of distinct elements is reduced almost by six and in the case $k = 4$ the storage ratio becomes almost 24. Since higher order tensors have very many entries, one has to utilize symmetric storage modes.

The draw-back of symmetric storage modes is that the access of individual elements is somewhat complicated, requiring for example three integer operations for address computations in the implementation of Berz [Ber91]. Moreover, the resulting memory locations may be far apart with irregular spacing so that significant paging overhead may be incurred. None of these difficulties arises when $n = 1$. Then for any intermediate value w the directional derivatives $w, \nabla_s w, \nabla_s^2 w, \ldots, \nabla_s^d w$ can be stored and manipulated as a contiguous array of $(d+1)$ scalars. Here d denotes the highest degree that is desired. It will be shown in Section 4 that to reconstruct the full tensors $\nabla_S^k f(\boldsymbol{x})$ for $k = 0, \ldots, d$, exactly as many univariate Taylor series are needed as the highest tensor $\nabla_S^d f(\boldsymbol{x})$ has distinct elements. As one will see, that entails a slight increase in storage, but a very significant gain in code simplicity and efficiency. After this outlook, the manipulation of Taylor polynomials is considered. The collection of Taylor coefficients that constitute the tensors

$$\nabla_S^k f(\boldsymbol{x}) \quad , \quad k = 0, \ldots, d,$$

represents a general polynomial of degree d in n variables. As can be concluded from (1), $\nabla_S^k f(\boldsymbol{x})$ contains

$$\binom{n+d}{d} = \sum_{k=0}^{d} \binom{n+k-1}{k} \tag{2}$$

distinct monomials. The truncated Taylor polynomials of all intermediate scalar quantities w have exactly the same structure. Considering again a product $w = u * v$, for the computation of $\nabla_S^k w$ each element of $\nabla_S^j u$, $0 \le j \le k$, has to be multiplied with all elements of $\nabla_S^{k-j} v$. It follows that

$$\binom{2n+k-1}{k} = \sum_{j=0}^{k} \binom{n+j-1}{j} \binom{n+k-j-1}{k-j}$$

multiplications are necessary because of (1). Hence, the total count of multiplications for computing $\nabla_S^d w$ is

$$\binom{2n+d}{d} = \sum_{k=0}^{d} \binom{2n+k-1}{k}.$$

When $n > 1$ there is also a significant number of additions and other overhead for each multiplication. Nevertheless it is possible to consider the number

$$\binom{2n+d}{d} \approx \frac{1}{d!}(2n)^d \quad \text{if} \quad d \ll n$$

as a reasonable approximation for the factor by which the cost to evaluate f grows when the calculation is performed in Taylor arithmetic of degree d and order n. Here, the fact is tacitly used that propagating Taylor polynomials through nonlinear functions such as the exponential, logarithm, and trigonometric functions is about as costly as the convolution for the product discussed above. All linear operations are cheaper, of course, since for them the effort is roughly $\binom{n+d}{d}$ and thus about the same as the number of data entries that need to be fetched and stored from and into memory.

Provided there is a significant fraction of nonlinear elementary functions in the overall calculation, one may consider

$$\binom{2n+d}{d} \Big/ \binom{n+d}{d} \approx 2^d$$

as an approximate computation/communication ratio for propagating Taylor polynomials. Consequently, even on a modern super-scalar processor with comparatively slow memory access, communication should not be the bottle-neck.

Now suppose that instead of propagating one Taylor polynomial in n variables and degree d one propagates $\binom{n+d-1}{d}$ univariate Taylor polynomials of the same degree d. Since the common constant term needs to be stored only once, the amount of data per intermediate becomes

$$1 + \binom{n+d-1}{d} d \quad = \quad 1 + \frac{d\,n}{d+n} \binom{n+d}{d}. \tag{3}$$

By inspection of the right hand side we see this is at most d times that for the standard case (see (2)). Furthermore,

$$\frac{(d+2)(d+1)}{2} \quad = \quad \binom{d+2}{d}$$

multiplications are needed to propagate one univariate Taylor polynomial of degree d through an elementary multiplication. Hence, the total amount of the computational effort for propagating $\binom{n+d-1}{d}$ univariate Taylor polynomials of the degree d is essentially given by

$$\binom{n+d-1}{d} \binom{d+2}{d} \quad = \quad q(d,n) \binom{2n+d}{d}, \tag{4}$$

where

$$q(d,n) \quad \equiv \quad \frac{(d+2)(d+1)}{2} \cdot \frac{(n+d-1)\dots(n)}{(2n+d)(2n+d-1)\dots(2n+1)}.$$

It is easy to check through induction that q is never greater than $\frac{3}{2}$ and that

$$q(d) \quad \equiv \quad \lim_{n\to\infty} q(d,n) \quad = \quad \frac{(d+2)(d+1)}{2^{d+1}}.$$

One has in particular $q(0,n) = 1$ and

$$\begin{aligned} q(1,n) \quad &= \quad \tfrac{3}{2} - \tfrac{3}{4n+2} \quad &&= \quad \tfrac{3}{2} - \mathcal{O}(\tfrac{1}{n}) \\ q(2,n) \quad &= \quad \tfrac{3}{2} - \tfrac{3}{4n+2} \quad &&= \quad \tfrac{3}{2} - \mathcal{O}(\tfrac{1}{n}) \\ q(3,n) \quad &= \quad \tfrac{5n(n+2)}{(2n+3)(2n+1)} \quad &&= \quad \tfrac{5}{4} - \mathcal{O}(\tfrac{1}{n}) \end{aligned}$$

$$\begin{aligned} q(4,n) &= \tfrac{15n(n+3)}{4(2n+3)(2n+1)} &&= \tfrac{15}{16} - \mathcal{O}(\tfrac{1}{n}) \\ q(5,n) &= \tfrac{21n(n+3)(n+4)}{4(2n+5)(2n+3)(2n+1)} &&= \tfrac{21}{32} - \mathcal{O}(\tfrac{1}{n}) \end{aligned}$$

as well as for all higher derivatives $d \geq 6$

$$q(d,n) = \frac{(d+2)(d+1)}{2^{d+1}} + \mathcal{O}(\frac{1}{n}).$$

In other words the computational effort is dramatically reduced when the degree d is quite large. This fact can also be seen in Figure 1 that displays the function $q(d,n)$. For the

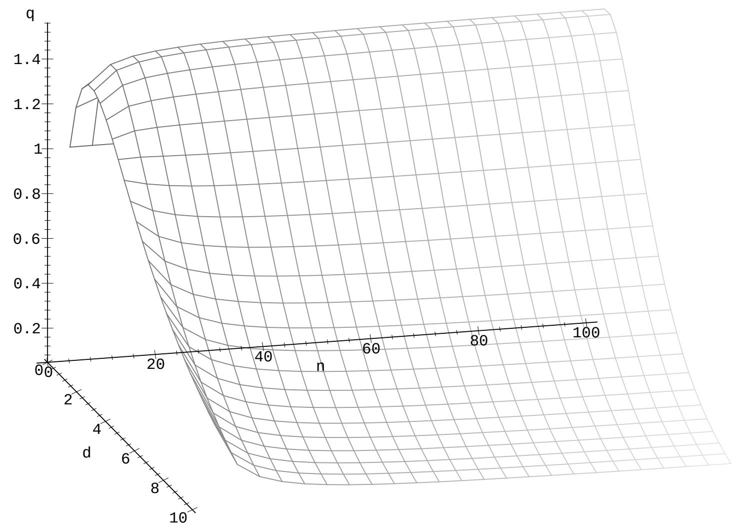

Figure 1: $q(d,n)$ for $n = 4,\ldots,100$, $\;d = 0,\ldots,10$

probably more important moderate values of $d \leq 5$ the complexity ratio is surprisingly close to one. The computations/communications ratio is about $d/2$ almost independent of n (see (3) and (4)). For $d \leq 4$ that ratio might appear rather small. However, the memory access is now strictly sequential, and no addressing calculations are needed.

In this section, we have demonstrated that for moderate values of d the propagation of $\binom{n+d-1}{d}$ univariate Taylor series has essentially the same operations count as multivariate derivative propagation. We found also that for higher degree derivatives, one needs considerable less operations using univariate Taylor series instead of the multivariate one. In the next section, an efficient scheme for interpolating all partial derivatives up to degree d from these $\binom{n+d-1}{d}$ univariate Taylor series is developed.

4 Interpolation with univariate Taylor series

For each direction $\boldsymbol{s} \in \mathbb{R}^{\bar{n}}$ one can obtain the Taylor expansion

$$f(\boldsymbol{x} + t\boldsymbol{s}) = f(\boldsymbol{x}) + f^{(1)}(\boldsymbol{x};\boldsymbol{s})t + f^{(2)}(\boldsymbol{x};\boldsymbol{s})t^2 + \cdots + f^{(d)}(\boldsymbol{x};\boldsymbol{s})t^d + \mathcal{O}(t^{d+1}). \quad (5)$$

The notation $f^{(m)}(\boldsymbol{x};\boldsymbol{s})$ will be used throughout to denote the m-th homogeneous polynomial in $\boldsymbol{s}$ in the Taylor expansion of f at $\boldsymbol{x}$. Hence, $\boldsymbol{x}$ is considered constant and $\boldsymbol{s} \in \mathbb{R}^n$ variable with the homogeneity property

$$f^{(m)}(\boldsymbol{x};t\boldsymbol{s}) \quad = \quad t^m f^{(m)}(\boldsymbol{x};\boldsymbol{s}) \qquad \text{for } t \in \mathbb{R}. \tag{6}$$

Since only the last coefficient $f^{(d)}(\boldsymbol{x};\boldsymbol{s})$ contains any information about the highest tensor $\nabla_{\boldsymbol{s}}^d f(\boldsymbol{x})$ with its $\binom{n+d-1}{d}$ distinct elements, it is clear that one needs to evaluate at least that many univariate expansions. We will see that this number is sufficient.

Let $\boldsymbol{i} = (i_1, \ldots, i_n) \in \mathbb{N}_0^n$ with $\boldsymbol{i}_j \in \mathbb{N}_0$ $(j = 1, \ldots, n)$ be a multi-index. The norm of $\boldsymbol{i}$ is defined by

$$|\boldsymbol{i}| = \sum_{j=1}^{n} \boldsymbol{i}_j.$$

Consider the matrix $\boldsymbol{S}$ throughout this section as fixed. Now choose the directions

$$\boldsymbol{S}\boldsymbol{i} \qquad \forall \boldsymbol{i} \in \mathbb{N}_0^n \text{ with } |\boldsymbol{i}| = d.$$

After selecting a particular d, Taylor coefficients are propagated along these $\binom{n+d-1}{d}$ necessary directions only. In other words, for each multi-index $\boldsymbol{i} \in \mathbb{N}_0^n$ with $|\boldsymbol{i}| = d$, we consider the Taylor series for the univariate function

$$\varphi_{\boldsymbol{i}} : \mathbb{R} \mapsto \mathbb{R}, \qquad \varphi_{\boldsymbol{i}}(t) \;\equiv\; f(\boldsymbol{x} + t\boldsymbol{S}\boldsymbol{i}),$$

and evaluate the Taylor coefficients of $\varphi_{\boldsymbol{i}}$ up to the degree d. The dependence of the $\varphi_{\boldsymbol{i}}$ on $\boldsymbol{S}$ is not denoted explicitly, because $\boldsymbol{S}$ is considered as fixed.

Note that there is no propagation along the directions $\boldsymbol{S}\boldsymbol{i}$ with $|\boldsymbol{i}| < d$. However, we will see that it is still possible to obtain all lower order derivative information that is necessary to compute the tensors $\nabla_{\boldsymbol{S}}^m f$ with $m < d$.

Interpolating Second Derivatives

To illustrate the approach presented here, let us first consider the computation of the Hessian $\nabla_{\boldsymbol{S}}^2 f$ with $\boldsymbol{S} = \boldsymbol{I}$ when the maximal degree d equals 2. Denote the i-th unit vector in $\mathbb{R}^n$ by $\boldsymbol{e}_i$. Then, the restricted gradient components of $\nabla_{\boldsymbol{S}} f(\boldsymbol{x})$ are obtained immediately as

$$\nabla_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_i \quad = \quad \frac{1}{2}\varphi_{\boldsymbol{i}}'(0) \quad = \quad \frac{1}{2}\nabla f(\boldsymbol{x})\boldsymbol{S}\boldsymbol{i} \quad \text{with} \quad \boldsymbol{i} \in \mathbb{N}_0^n,\; \boldsymbol{i}_k = \begin{cases} 2, & k = i \\ 0, & k \neq i \end{cases}. \tag{7}$$

Similarly, the pure second derivatives can be obtained from

$$\boldsymbol{e}_i^T \nabla_{\boldsymbol{S}}^2 f(\boldsymbol{x})\boldsymbol{e}_i \quad = \quad \frac{1}{4}\varphi_{\boldsymbol{i}}''(0) \quad = \quad \frac{1}{4}(\boldsymbol{S}\boldsymbol{i})^T \nabla^2 f(\boldsymbol{x})\boldsymbol{S}\boldsymbol{i}$$

with the same multi-index $\boldsymbol{i}$ as in (7). However, the mixed second derivatives $\boldsymbol{e}_i^T \nabla_{\boldsymbol{S}}^2 f(\boldsymbol{x})\boldsymbol{e}_j$, $i \neq j$ are not directly available. To get them, one has to consider the diagonal direction $\boldsymbol{S}\boldsymbol{l}$ defined by the multi-index

$$\boldsymbol{l} \in \mathbb{N}_0^n \quad \text{with} \quad \boldsymbol{l}_k = \begin{cases} 1, & k = i \text{ or } k = j \\ 0, & k \neq i \text{ and } k \neq j \end{cases}.$$

Because of $|\boldsymbol{l}| = 2$ the second order Taylor coefficient of $\varphi_{\boldsymbol{l}}$ is known

$$\varphi''_{\boldsymbol{l}}(0) = (\boldsymbol{e}_i + \boldsymbol{e}_j)^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})(\boldsymbol{e}_i + \boldsymbol{e}_j) = \boldsymbol{e}_i^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_i + \boldsymbol{e}_j^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_j + 2\boldsymbol{e}_i^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_j .$$

This identity yields the interpolation formula

$$\begin{aligned} \boldsymbol{e}_i^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_j &= \frac{1}{2}[(\boldsymbol{e}_i + \boldsymbol{e}_j)^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})(\boldsymbol{e}_i + \boldsymbol{e}_j) - \boldsymbol{e}_i^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_i - \boldsymbol{e}_j^T \nabla^2_{\boldsymbol{S}} f(\boldsymbol{x})\boldsymbol{e}_j] \\ &= \frac{1}{2}\varphi''_{\boldsymbol{l}}(0) - \frac{1}{8}\varphi''_{\boldsymbol{i}}(0) - \frac{1}{8}\varphi''_{\boldsymbol{j}}(0) \end{aligned} \tag{8}$$

with the multi-indices $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{l} \in \mathbb{N}_0^n$ and

$$\boldsymbol{i}_k = \begin{cases} 2, & k = i \\ 0, & k \neq i \end{cases}, \quad \boldsymbol{j}_k = \begin{cases} 2, & k = j \\ 0, & k \neq j \end{cases} \quad \text{and} \quad \boldsymbol{l}_k = \begin{cases} 1, & k = i \text{ or } k = j \\ 0, & k \neq i \text{ and } k \neq j \end{cases} . \tag{9}$$

When $d = 3$ this scheme is not directly applicable because $|\boldsymbol{l}| < 3$. Therefore $\boldsymbol{l}$ does not aim at a derivative of the highest order and one does not propagate a Taylor series along the direction $\boldsymbol{S}\boldsymbol{l}$. Hence, $\varphi''_{\boldsymbol{l}}(0)$ is unknown. To handle that situation and generally the higher order cases one needs a systematic way of generating formula of the form (8) for mixed partials.

Computing Mixed Partials from Grid Values

For any polynomial P of degree n or less, it can be checked that

$$\frac{\partial^n P(\boldsymbol{S}\boldsymbol{z})}{\partial z_1 \partial z_2 \ldots \partial z_n} = \sum_{i_1=0}^{1} \cdots \sum_{i_n=0}^{1} P(i_1 \boldsymbol{s}_1 + \cdots + i_n \boldsymbol{s}_n)(-1)^{n-(i_1+\cdots+i_n)} \tag{10}$$

through integration of the constant function on the left side over the unit cube in n dimensions. The important observation here is that one only has to know the value of the polynomial at the corners of the parallelepiped $\{\boldsymbol{S}\boldsymbol{z} : 0 \leq \boldsymbol{z}_i \leq 1\}$ in order to compute the mixed derivative with respect to all $\boldsymbol{z}_i$. Naturally, one does not want to assume that f itself is a polynomial. However, one can use the m-th coefficients of the univariate Taylor expansions to compute values of the homogeneous approximating polynomials $f^{(m)}(\boldsymbol{x}; \boldsymbol{s})$, $1 \leq m \leq d$, satisfying (5) and (6). For example it follows immediately for the second mixed partials with $P(\boldsymbol{S}\boldsymbol{z}) = f^{(2)}(\boldsymbol{x}; \boldsymbol{S}\boldsymbol{z})$ and $1 \leq j < i \leq n$

$$\frac{\partial^2 f^{(2)}(\boldsymbol{x}; \boldsymbol{S}\boldsymbol{z})}{\partial z_i \partial z_j} = \underbrace{f^{(2)}(\boldsymbol{x}; \boldsymbol{S}(\boldsymbol{e}_i + \boldsymbol{e}_j))}_{=\frac{1}{2}\varphi''_{\boldsymbol{l}}(0)} - \underbrace{f^{(2)}(\boldsymbol{x}; \boldsymbol{S}\boldsymbol{e}_i)}_{=\frac{1}{8}\varphi''_{\boldsymbol{i}}(0)} - \underbrace{f^{(2)}(\boldsymbol{x}; \boldsymbol{S}\boldsymbol{e}_j)}_{=\frac{1}{8}\varphi''_{\boldsymbol{j}}(0)} + f^{(2)}(\boldsymbol{x}; 0) \tag{11}$$

with the multi-indices $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{l}$ defined as in (9). Through the homogeneity property (6) one obtains that

$$f^{(m)}(\boldsymbol{x}; 0) = 0, \qquad \forall\, m > 0, \tag{12}$$

so that the last term in (11) drops out and one gets

$$\frac{\partial^2 f^{(2)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{z})}{\partial z_i \partial z_j} \equiv \left.\frac{\partial^2 f(\boldsymbol{x}+\boldsymbol{S}\boldsymbol{z})}{\partial z_i \partial z_j}\right|_{\boldsymbol{z}=0}$$

because of the previous formula (8). For fixed $\boldsymbol{S}$ and given d, the real numbers $f^{(|\boldsymbol{i}|)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{i})$ with $|\boldsymbol{i}| \leq d$ will be referred to as the grid values.

By considering seed matrices $\boldsymbol{S} \in \mathbb{R}^{\bar{n}\times d}$ with repeated columns one can derive from (10) for any polynomial P of degree up to d its generalization

$$\frac{\partial^{|\boldsymbol{i}|}P(\boldsymbol{S}\boldsymbol{z})}{\partial z_1^{i_1}\partial z_2^{i_2}\ldots\partial z_n^{i_n}} = \sum_{k_1=0}^{i_1}\sum_{k_2=0}^{i_2}\cdots\sum_{k_n=0}^{i_n}\binom{i_1}{k_1}\binom{i_2}{k_2}\cdots\binom{i_n}{k_n}(-1)^{|\boldsymbol{i}-\boldsymbol{k}|}P(\boldsymbol{S}\boldsymbol{k}),$$

where now $\boldsymbol{S} \in \mathbb{R}^{\bar{n}\times n}$ and $\boldsymbol{i}$ may be any multi-index with $|\boldsymbol{i}| = deg(P)$. Applying this identity to the homogeneous components $f^{(|\boldsymbol{i}|)}(\boldsymbol{x};\boldsymbol{s})$ of f at $\boldsymbol{x}$ and using binomial coefficient notation for multi-indices one obtains with (12) and the zero vector $\mathbf{0} \in \mathbb{N}_0^n$

$$\boxed{f_{\boldsymbol{i}}(\boldsymbol{x}) \equiv \left.\frac{\partial^{|\boldsymbol{i}|}f(\boldsymbol{x}+\boldsymbol{S}\boldsymbol{z})}{\partial z_1^{i_1}\partial z_2^{i_2}\ldots\partial z_n^{i_n}}\right|_{\boldsymbol{z}=0} = \sum_{\mathbf{0}<\boldsymbol{k}\leq\boldsymbol{i}}\binom{\boldsymbol{i}}{\boldsymbol{k}}(-1)^{|\boldsymbol{i}-\boldsymbol{k}|}f^{(|\boldsymbol{i}|)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{k}).} \quad (13)$$

We conclude this subsection with a geometric illustration of the situation. Suppose one wishes to obtain the partials $f_{\boldsymbol{i}}$ for all $\boldsymbol{i}$ with $1 \leq |\boldsymbol{i}| \leq d$. The functions $f^{(m)}(\boldsymbol{x};\boldsymbol{s})$, $\boldsymbol{s} \in \mathbb{R}^{\bar{n}}$, denote the m-th homogeneous polynomial at $\boldsymbol{x}$ as in equation (5). Through formula (13) it is possible to calculate $f_{\boldsymbol{i}}(\boldsymbol{x})$ from the grid values $f^{(|\boldsymbol{i}|)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{k})$ for all $\boldsymbol{k} \leq \boldsymbol{i}$. However, the propagation of univariate Taylor series only yields some of the grid values directly. The others must be calculated in a second interpolation step. The values $f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{i})$ are known for all $|\boldsymbol{i}| = d$ and for all $0 \leq m \leq d$. They will be referred to as ray values. The Figure 2 illustrates this situation for $n = 2$ and $d = 4$. The thin lines represent the directions of the

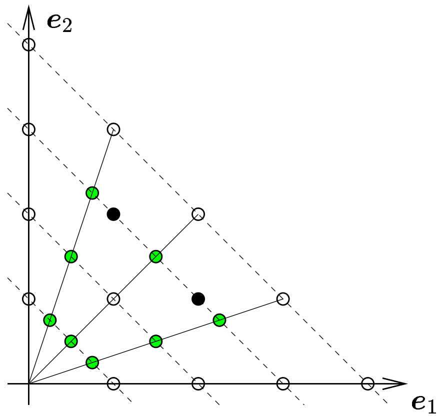

Figure 2: Grid values and ray values for $n = 2$ and $d = 4$

propagated Taylor series including the $\boldsymbol{e}_1$- and the $\boldsymbol{e}_2$-axes. The grid points whose values can be obtained by scaling of the ray values $f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{i})$ with $|\boldsymbol{i}| = d$ and $0 \le m \le d$ are marked by unfilled circles. The black circles denote the values that are desired because they arise on the right hand side of (13) but are as yet unknown. The grey shaded circles mark points that do not belong to the grid but whose values can be obtained by scaling.

The following section describes the way to compute the unknown grid values from the known ray values.

Computing Grid Values from Ray Values

For all multi-indices $\boldsymbol{k}$ with $|\boldsymbol{k}| < d$, the fact that $f^{(m)}(\boldsymbol{x};\boldsymbol{s})$ is homogeneous of degree m implies

$$f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{k}) \;=\; (|\boldsymbol{k}|/d)^m f^{(m)}\left(\boldsymbol{x};\boldsymbol{S}(d\boldsymbol{k}/|\boldsymbol{k}|)\right). \tag{14}$$

The ray values are placed at equal distance and the polynomial $\binom{\boldsymbol{z}}{\boldsymbol{j}} f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{j})$ in z is nonzero for only one ray point. Hence, an interpolation process similar to the one dimensional Lagrange interpolation formula yields the equation

$$f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{z}) \;=\; \sum_{|\boldsymbol{j}|=d} \binom{\boldsymbol{z}}{\boldsymbol{j}} f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{j}) \tag{15}$$

for $\boldsymbol{z} = d\boldsymbol{k}/|\boldsymbol{k}|$ or any other vector $\boldsymbol{z} \in \mathbb{R}^n$ with value $|\boldsymbol{z}| = d$. Therefore the ray values $f^{(m)}\left(\boldsymbol{x};\boldsymbol{S}(d\boldsymbol{k}/|\boldsymbol{k}|)\right)$ can be computed for any $\boldsymbol{k}$ with $|\boldsymbol{k}| < d$. Substituting (15) into (14) one obtains

$$\boxed{f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{k}) \;=\; \sum_{|\boldsymbol{j}|=d} \left(\frac{|\boldsymbol{k}|}{d}\right)^m \binom{d\boldsymbol{k}/|\boldsymbol{k}|}{\boldsymbol{j}} f^{(m)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{j}).} \tag{16}$$

Direct Computation of Mixed Partials from Ray Values

Now the main result of this article can be established, namely an efficient scheme for calculating all mixed partial derivatives up to degree d from the known ray values. The following proposition contains the explicit formula and an estimation of the complexity of the interpolation procedure.

Proposition:
Let f be at least d-times continuously differentiable at some point $\boldsymbol{x} \in \mathbb{R}^{\bar{n}}$. Then for any $\boldsymbol{i} \in \mathbb{N}_0^n$ with $1 \le |\boldsymbol{i}| \le d$ the partial derivative $f_{\boldsymbol{i}}(\boldsymbol{x})$ defined in (13) is given by the formula

$$\boxed{f_{\boldsymbol{i}}(\boldsymbol{x}) \;=\; \sum_{|\boldsymbol{j}|=d} f^{(|\boldsymbol{i}|)}(\boldsymbol{x};\boldsymbol{S}\boldsymbol{j}) \underbrace{\sum_{\boldsymbol{0}<\boldsymbol{k}\le\boldsymbol{i}} (-1)^{|\boldsymbol{i}-\boldsymbol{k}|} \binom{\boldsymbol{i}}{\boldsymbol{k}} \binom{d\boldsymbol{k}/|\boldsymbol{k}|}{\boldsymbol{j}} \left(\frac{|\boldsymbol{k}|}{d}\right)^{|\boldsymbol{i}|}}_{\equiv\, c_{\boldsymbol{i},\boldsymbol{j}}}} \tag{17}$$

The number of nonvanishing coefficients

$$0 \neq c_{\boldsymbol{i},\boldsymbol{j}} \qquad \textit{for } \boldsymbol{i},\boldsymbol{j} \in \mathbb{N}_0^n,\ 1 \leq |\boldsymbol{i}| \leq d,\ |\boldsymbol{j}| = d, \tag{18}$$

is less than or equal to

$$p(d,n) \equiv \sum_{m=1}^{d} \binom{n}{m}\binom{d}{m}\binom{m+d-1}{d},$$

which bounds the number of operations needed to calculate all partial derivatives from the univariate Taylor series.

Proof:
Combining (13) and (16) one obtains the identity (17), which can be written in a simpler form as

$$f_{\boldsymbol{i}}(\boldsymbol{x}) = \sum_{|\boldsymbol{j}|=d} c_{\boldsymbol{i},\boldsymbol{j}} f^{(|\boldsymbol{i}|)}(\boldsymbol{x}; \boldsymbol{S}\boldsymbol{j}). \tag{19}$$

Define the sign function for multi-indices componentwise as follows

$$\mathbf{sign}(\boldsymbol{i}) \equiv (\mathrm{sign}(\boldsymbol{i}_1), \ldots, \mathrm{sign}(\boldsymbol{i}_n)) \qquad \forall\, \boldsymbol{i} \in \mathbb{N}_0^n.$$

For two multi-indices $\boldsymbol{i}$ and $\boldsymbol{j}$ with $1 \leq |\boldsymbol{i}| \leq d$ and $|\boldsymbol{j}| = d$, the coefficient $c_{\boldsymbol{i},\boldsymbol{j}}$ can only be nonzero if

$$\mathbf{sign}(\boldsymbol{j}) \leq \mathbf{sign}(\boldsymbol{i}) \tag{20}$$

since otherwise the second binomial coefficient in (17) must vanish for all $\mathbf{0} < \boldsymbol{k} \leq \boldsymbol{i}$.

Now, consider an arbitrary $m \in \mathbb{N}$ with $1 \leq m \leq d$. A multi-index $\boldsymbol{i}$ with $|\mathbf{sign}(\boldsymbol{i})| = m$ has m nonvanishing components. Therefore the inequality

$$m \leq |\boldsymbol{i}| \leq d$$

must hold for all $\boldsymbol{i}$ with $|\mathbf{sign}(\boldsymbol{i})| = m$. The number of distinct possibilities to choose m positive integers so that their sum is no greater than d is given by $\binom{d}{m}$.

It follows that the total number of multi-indices $\boldsymbol{i}$ with $1 \leq |\boldsymbol{i}| \leq d$ and $|\mathbf{sign}(\boldsymbol{i})| = m$ is equal to $\binom{n}{m}\binom{d}{m}$. For each of these multi-indices the coefficient $c_{\boldsymbol{i},\boldsymbol{j}}$ can only be greater than zero if $\boldsymbol{j}$ satisfies the necessary condition (20). This implies

$$|\mathbf{sign}(\boldsymbol{j})| \leq m.$$

Therefore, one can conclude that for each multi-index $\boldsymbol{i}$ with $1 \leq |\boldsymbol{i}| \leq d$ and $|\mathbf{sign}(\boldsymbol{i})| = m$ the number of of multi-indices $\boldsymbol{j}$ with $\mathbf{sign}(\boldsymbol{j}) \leq \mathbf{sign}(\boldsymbol{i})$ is given by $\binom{m+d-1}{d}$.

Hence we conclude that the function

$$p(d,n) \equiv \sum_{m=1}^{d} \binom{n}{m}\binom{d}{m}\binom{m+d-1}{d} \tag{21}$$

denotes an upper bound for the number of nonvanishing coefficients $c_{\boldsymbol{i},\boldsymbol{j}}$. One can derive from (19) that (21) is also an upper bound for the total number of multiplications of the interpolation procedure. ∎

It follows from the tables at the end of this section that for $d = 2, 3$ and any $n \in \mathbb{N}$, the function $p(d,n)$ gives the exact number of nonvanishing coefficients $c_{\boldsymbol{i},\boldsymbol{j}}$. Therefore it may be conjectured that $p(d,n)$ corresponds exactly to the number of nonvanishing factors $c_{\boldsymbol{i},\boldsymbol{j}}$, but so far there is no proof of this. Define

$$r(d,n) \;\equiv\; \frac{p(d,n)}{\binom{2n+d}{d}} \;=\; \frac{1}{2^d}\binom{2d-1}{d} + \mathcal{O}\left(\frac{c_d}{n}\right)$$

as the ratio of the upper bound $p(d,n)$ and the complexity $\binom{2n+d}{d}$ of propagating multivariate Taylor series through a single multiplication operation. Hence, one obtains for large d

$$r(d,n) \;\approx\; \frac{2^{d-1}}{\sqrt{\pi d}} + \mathcal{O}\left(\frac{c_d}{n}\right).$$

As can be seen in Figure 3 the ratio $r(d,n)$ is quite small for the more important moderate

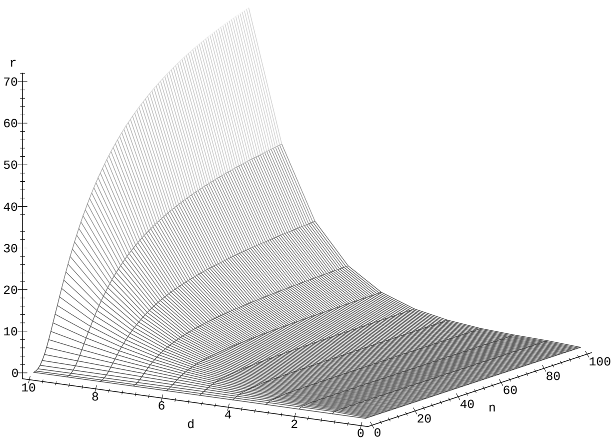

Figure 3: $r(d,n)$ for $n = 1,\ldots,100,\;\; d = 0,\ldots,10$

values $d \le 5$. It follows in particular

$$\begin{aligned}
r(1,n) &= \tfrac{1}{2} - \tfrac{1}{4n+2} &&= \tfrac{1}{2} + \mathcal{O}(\tfrac{1}{n})\\
r(2,n) &= \tfrac{n(1+3n)}{2(2n+1)(n+1)} &&= \tfrac{3}{4} + \mathcal{O}(\tfrac{1}{n})\\
r(3,n) &= \tfrac{n(1+3n+5n^2)}{(2n+3)(2n+1)(n+1)} &&= \tfrac{5}{4} + \mathcal{O}(\tfrac{1}{n})
\end{aligned}$$

as well as

$$\begin{aligned}
r(4,n) &= \tfrac{35}{16} + \mathcal{O}(\tfrac{1}{n}) &&\approx 2 + \mathcal{O}(\tfrac{1}{n})\\
r(5,n) &= \tfrac{63}{16} + \mathcal{O}(\tfrac{1}{n}) &&\approx 4 + \mathcal{O}(\tfrac{1}{n})\\
r(6,n) &= \tfrac{231}{32} + \mathcal{O}(\tfrac{1}{n}) &&\approx 7 + \mathcal{O}(\tfrac{1}{n})\\
r(7,n) &= \tfrac{429}{32} + \mathcal{O}(\tfrac{1}{n}) &&\approx 13 + \mathcal{O}(\tfrac{1}{n}).
\end{aligned}$$

To apply the formula (19) at various points $\boldsymbol{x} \in \mathbb{R}^{\bar{n}}$ it makes sense to precompute the rational coefficients (18). So far a simple, explicit formula for them has not been found, but it is possible to further reduce the computation by grouping the additive terms together. A listing is given below for $d = 2$ and $d = 3$.

When $d = 2$ one has to consider the following four groups of multi-indices

$$\begin{array}{lcl} M_0 & = & \{\mathbf{0} \in \mathbb{N}_0^n\} \\ M_1 & = & \{\boldsymbol{i} \in \mathbb{N}_0^n \mid |\boldsymbol{i}| = 1\} \\ M_2 & = & \{\boldsymbol{i} \in \mathbb{N}_0^n \mid |\boldsymbol{i}| = 2 \text{ and } |\mathbf{sign}(\boldsymbol{i})| = 1\} \\ M_3 & = & \{\boldsymbol{i} \in \mathbb{N}_0^n \mid |\boldsymbol{i}| = 2 \text{ and } |\mathbf{sign}(\boldsymbol{i})| = 2\} \end{array} \tag{22}$$

Then, the coefficient $c_{\boldsymbol{i},\boldsymbol{j}}$ has to be calculated for each $\boldsymbol{i} \in M_0 \cup M_1 \cup M_2 \cup M_3$ and each $\boldsymbol{j} \in M_2 \cup M_3$. The Table 1 lists the nonvanishing $c_{\boldsymbol{i},\boldsymbol{j}}$.

Multi-indices			$c_{\boldsymbol{i},\boldsymbol{j}}$
$\boldsymbol{i} \in M_1$	$\boldsymbol{j} \in M_2$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$\frac{1}{2}$
$\boldsymbol{i} \in M_2$	$\boldsymbol{j} \in M_2$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$\frac{1}{2}$
$\boldsymbol{i} \in M_3$	$\boldsymbol{j} \in M_2$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$-\frac{1}{4}$
$\boldsymbol{i} \in M_3$	$\boldsymbol{j} \in M_3$	$\boldsymbol{i}^T\boldsymbol{j} = 2$	1

Table 1: Nonvanishing coefficients for $d = 2$

For $d = 3$ there are three groups of multi-indices besides those of (22)

$$\begin{array}{lcl} M_4 & = & \{\boldsymbol{i} \in \mathbb{N}_0^n \mid |\boldsymbol{i}| = 3 \text{ and } |\mathbf{sign}(\boldsymbol{i})| = 1\} \\ M_5 & = & \{\boldsymbol{i} \in \mathbb{N}_0^n \mid |\boldsymbol{i}| = 3 \text{ and } |\mathbf{sign}(\boldsymbol{i})| = 2\} \\ M_6 & = & \{\boldsymbol{i} \in \mathbb{N}_0^n \mid |\boldsymbol{i}| = 3 \text{ and } |\mathbf{sign}(\boldsymbol{i})| = 3\} \end{array}$$

Now, for each $\boldsymbol{i} \in M_0 \cup \ldots \cup M_6$ and each $\boldsymbol{j} \in M_4 \cup M_5 \cup M_6$ the coefficients (18) must be considered. The nonvanishing $c_{\boldsymbol{i},\boldsymbol{j}}$ are given by Table 2.

Multi-indices			$c_{\boldsymbol{i},\boldsymbol{j}}$	Multi-indices			$c_{\boldsymbol{i},\boldsymbol{j}}$
$\boldsymbol{i} \in M_1$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$\frac{1}{3}$	$\boldsymbol{i} \in M_5$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} = 3$	$\frac{2}{27}$
$\boldsymbol{i} \in M_2$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$\frac{2}{9}$	$\boldsymbol{i} \in M_5$	$\boldsymbol{j} \in M_5$	$\boldsymbol{i}^T\boldsymbol{j} = 5$	$\frac{2}{3}$
$\boldsymbol{i} \in M_3$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$-\frac{5}{36}$	$\boldsymbol{i} \in M_5$	$\boldsymbol{j} \in M_5$	$\boldsymbol{i}^T\boldsymbol{j} = 4$	$-\frac{1}{3}$
$\boldsymbol{i} \in M_3$	$\boldsymbol{j} \in M_5$	$\boldsymbol{i}^T\boldsymbol{j} = 3$	$\frac{1}{4}$	$\boldsymbol{i} \in M_6$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$\frac{2}{27}$
$\boldsymbol{i} \in M_4$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} \neq 0$	$\frac{2}{9}$	$\boldsymbol{i} \in M_6$	$\boldsymbol{j} \in M_5$	$\boldsymbol{i}^T\boldsymbol{j} = 3$	$-\frac{1}{6}$
$\boldsymbol{i} \in M_5$	$\boldsymbol{j} \in M_4$	$\boldsymbol{i}^T\boldsymbol{j} = 6$	$-\frac{5}{27}$	$\boldsymbol{i} \in M_6$	$\boldsymbol{j} \in M_6$	$\boldsymbol{i}^T\boldsymbol{j} = 3$	1

Table 2: Nonvanishing coefficients for $d = 3$

As one can see all coefficients $c_{\boldsymbol{i},\boldsymbol{j}}$ for $|d| \leq 3$ are of moderate size and most are positive. It should also be noted that the interpolation procedure does not involve any divisions so that it does not expand errors occurred by propagating the univariate Taylor series.

5 Some Run Time Results

This section presents some run time results for computing higher derivative tensors using the interpolation with univariate Taylor expansions described above.

To optimize bevel gears exact knowledge of the geometric properties of bevel gear tooth flanks is necessary (see e.g. [Hut96]). Arbitrary points on a given grid on the tooth flank can be calculated with machine accuracy by an analytical characterization [GR89] and computational differentiation. To that end one needs to compute the higher derivative tensors of a vector function $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^3$ describing a family of surfaces. A corresponding evaluation code consisting of approximately 160 C statements was differentiated using ADOL-C [GJU96] and a special driver for the calculation of higher derivative tensors by forward propagation of univariate Taylor series.

The run times observed on a Sun Sparc 10, always normalized by time for evaluating $\boldsymbol{f}$ without derivatives, are listed in the Table 3.

d	distinct elements in $\nabla^k \boldsymbol{f}(\boldsymbol{x})$, $k = 0, \ldots, d$	theoretical run time ratio	ratio for evaluating $\nabla^k \boldsymbol{f}(\boldsymbol{x})$, $k = 0, \ldots, d$	ratio for the interpolation
2	10	36	16.2	0.6
3	20	100	45.0	1.9
4	35	225	93.3	4.0
5	56	441	184.7	8.5
6	84	672	356.0	15.1
7	120	1290	655.3	26.3
8	165	2025	1174.0	44.4
9	220	3025	2040.7	81.0

Table 3: Run-time ratios for derivative tensor evaluations

The third column states the theoretical run time ratios given by $(d+2)^2(d+1)^2/4$ (see equation (4) in Section 3). The next column contains the ratios of the complete calculation of $\nabla^k \boldsymbol{f}(\boldsymbol{x})$, $k = 0, \ldots, d$, using ADOL-C and the special driver mentioned above. As can be seen the total ratio is about 50 % – 70 % of the theoretical estimate. The ratio of the run time of the interpolation process only and the run time of the function evaluation is contained in the fourth column. In agreement with the asymptotic expansion for $r(d, n)$, one observes a doubling of the ratio with each increment of d by 1. Nevertheless, one finds on this example for $d \leq 9$ that no more than five percent of the run time for the derivative calculation are needed to interpolate the coefficients of the derivative tensors from the univariate Taylor series.

6 Conclusions

Many applications in scientific computation require higher order derivatives. This article described a promising approach to compute higher order derivatives by using univariate Taylor series. The main result is equation (17), which represents general partial derivatives in terms of univariate Taylor coefficients.

It was found that the post-processing effort for the interpolation given by the function $p(d, n)$ in Section 4 is very small, especially for the more important moderate values of d.

Also, for these d the complexity ratio $q(d, n)$ analyzed in Section 3 between the new univariate Taylor approach and a more conventional multivariate Taylor approach is essentially 1. In addition the data structures and memory access pattern are much simpler and more regular so that actual run-times should be significantly reduced.

References

[BCG93] C. H. Bischof, G. F. Corliss, and A. Griewank. Structured second- and higher-order derivatives through univariate Taylor series. *Optimization Methods and Software*, 2:211–232, 1993.

[Ber89] M. Berz. Differential algebraic description of beam dynamics to very high orders. *Particle Accelerators*, 12:109–124, 1989.

[Ber91] M. Berz. Algorithms for higher derivatives in many variables with applications to bear physics. In G. F. Corliss and A. Griewank, editors, *Automatic Differentiation of Algorithms - Theory, Implementation, and Applications*, pages 147–156. SIAM, Philadelphia, Penn., 1991.

[GJU96] A. Griewank, D. Juedes, and J. Utke. ADOL-C, A package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, 22:131–167, 1996.

[GR89] A. Griewank and G. W. Reddien. Computation of cusp singularities for operator equations and their discretizations. *J. Comput. Appl. Math.*, pages 133–153, 1989.

[Gri89] A. Griewank. On automatic differentiation. In *Mathematical Programming: Recent Developments and Applications*, pages 83–108, Amsterdam, 1989. Kluwer Academic Publishers.

[Gri91] A. Griewank. Automatic evaluation of first- and higher-derivative vectors. In R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, editors, *Proceedings of the Conference "Bifurcation and Chaos: Analysis, Algorithms, Applications at Würzburg*, pages 124–137, Basel, 1991. Birkhäuser.

[Gri94] A. Griewank. Computational differentiation and optimization. In J. R. Birge and K. G. Murty, editors, *Mathematical Programming: State of the Art 1994*, pages 102–131. The University of Michigan, Michigan, USA, 1994.

[Hut96] U. Hutschenreiter. A new method for bevel gear tooth flank computation. In M. Berz, Chr. Bischof, G. Corliss, and A. Griewank, editors, *Computational Differentiation - Techniques, Applications, and Tools*, pages 161–172. SIAM, Philadelphia, Penn., 1996.

[Nei92] R. D. Neidinger. An efficient method for the numerical evaluation of partial derivatives of arbitrary order. *ACM Trans. Math. Software*, 18:159–173, 1992.