Stat Comput (2015) 25:959–974
DOI 10.1007/s11222-014-9465-1

Efficient recursive algorithms for functionals based on higher order derivatives of the multivariate Gaussian density

José E. Chacón · Tarn Duong

Received: 8 November 2013 / Accepted: 24 March 2014 / Published online: 13 April 2014

Abstract Many developments in Mathematics involve the computation of higher order derivatives of Gaussian density functions. The analysis of univariate Gaussian random variables is a well-established field whereas the analysis of their multivariate counterparts consists of a body of results which are more dispersed. These latter results generally fall into two main categories: theoretical expressions which reveal the deep structure of the problem, or computational algorithms which can mask the connections with closely related problems. In this paper, we unify existing results and develop new results in a framework which is both conceptually cogent and computationally efficient. We focus on the underlying connections between higher order derivatives of Gaussian density functions, the expected value of products of quadratic forms in Gaussian random variables, and V-statistics of degree two based on Gaussian density functions. These three sets of results are combined into an analysis of non-parametric data smoothers.

Electronic supplementary material The online version of this article (doi:10.1007/s11222-014-9465-1) contains supplementary material, which is available to authorized users.

J. E. Chacón (✉)
Departamento de Matemáticas, Universidad de Extremadura, 06006 Badajoz, Spain
e-mail: jechacon@unex.es

T. Duong
Theoretical and Applied Statistics Laboratory (LSTA), Sorbonne Universities, University Pierre and Marie Curie (UPMC) – Paris 6, UR 1, 75005 Paris, France
e-mail: tarn.duong@upmc.fr

T. Duong
Pitié-Salpêtrière Hospital, Institute of Translational Neurosciences (IHU-A-ICM), Assistance Publique-Hôpitaux de Paris (AP-HP), 75005 Paris, France

Keywords Hermite polynomial · Derivative · Kernel estimator · Normal density · Quadratic forms · Symmetrizer matrix · V-statistics

Mathematics Subject Classification 15A24 · 65F30 · 62E10 · 62G05 · 62H05

1 Introduction

Gaussian random variables and their associated probability density functions are commonly studied in Statistics since they possess many attractive theoretical and computational properties. In fact, Gaussian functions and its derivatives appear as fundamental tools in many areas of Mathematics, and also in other disciplines like Physics or Engineering.

Many results have been established for univariate Gaussian random variables in a unified framework. For multivariate random variables, many results are available as well, but due to the lack of a commonly accepted notation for higher order derivatives of the multivariate functions, these are more scattered.

In this paper we adopt the vectorized form of higher order multidimensional derivatives, which was the key tool that allowed to obtain explicit formulas for the moments of the multivariate normal distribution of arbitrary order (Holmquist 1988) and for the higher order derivatives of the multivariate Gaussian density function, through the introduction of vector Hermite polynomials (Holmquist 1996a).

These polynomials, however, depend on a matrix (so-called symmetrizer matrix) having an enormous number of entries, even when the dimension and the derivative order are not high. Thus, although these results provide a general formulation that is valid and useful for developing theory in any dimension and for an arbitrary derivative/moment order,

some authors like Triantafyllopoulos (2003) or Kan (2008) pointed out the difficulties that the computation of such a large matrix represents in practical situations.

Here we unify existing results, as well as developing new ones, in a cogent framework which facilitates a concise theoretical form as well as an efficient computational form. We begin by introducing the vectorized form of the higher order derivatives of the multivariate Gaussian density functions, via their factorization as Hermite polynomials, in Sect. 2. Efficient recursive algorithms to compute the involved high-dimensional symmetrizer matrix, and the product of this matrix and a high-dimensional vector, are discussed in Sects. 3 and 4, respectively. A different approach is developed in Sect. 5, by focusing on the recursive computation of the unique partial derivative operators that unequivocally determine the full derivative vector. We then focus on some statistical applications intimately linked with the derivatives of the multivariate Gaussian density function: the computation of moments of multivariate normal distributions and the expectation of powers of quadratic forms in Gaussian random variables are explored in Sects. 6.1 and 6.2, respectively, including a new result providing a formula for the joint cumulants of quadratic forms in normal variables that corrects an identity included in Mathai and Provost (1992). In Sect. 6.3 we show how these functionals are extremely useful for the analysis of non-parametric data smoothers, which involve the computation of V-statistics of degree two based on derivatives of multivariate Gaussian density functions. Finally, in Sect. 7 all the newly introduced recursive algorithms are compared to the standard, direct approach, in terms of computation time.

2 Higher order derivatives of Gaussian density functions

The characterization of the r-th order derivatives of a d-variate function can be expressed in many ways using, e.g. matrices, tensors or iterated permutations. We use the characterization using Kronecker products of vectors, popularized by Holmquist (1996a). Let f be a real d-variate function, $\boldsymbol{x} = (x_1, \ldots, x_d)$, and $\mathsf{D} = \partial/\partial\boldsymbol{x} = (\partial/\partial x_1, \ldots, \partial/\partial x_d)$ be the first derivative (gradient) operator. If the usual convention $(\partial/\partial x_i)(\partial/\partial x_j) = \partial^2/(\partial x_i \partial x_j)$ is taken into account, then the r-th derivative of f is defined to be the vector $\mathsf{D}^{\otimes r} f = (\mathsf{D} f)^{\otimes r} = \partial^r f/\partial \boldsymbol{x}^{\otimes r} \in \mathbb{R}^{d^r}$, with $\mathsf{D}^{\otimes 0} f = f$, $\mathsf{D}^{\otimes 1} f = \mathsf{D} f$. Here, $\mathsf{D}^{\otimes r}$ refers to the r-th Kronecker power of the operator D, formally understood as the r-fold product $\mathsf{D} \otimes \cdots \otimes \mathsf{D}$.

For example, all the second order partial derivatives can be organized into the usual Hessian matrix $\mathsf{H} f = (\partial^2 f/\partial x_i \partial x_j)_{i,j=1}^d$, and the Hessian operator can be formally written as $\mathsf{H} = \mathsf{D}\mathsf{D}^\top$. The equivalent vectorized form is $\mathsf{D}^{\otimes 2} = \operatorname{vec} \mathsf{H}$, where vec denotes the operator which concatenates the columns of a matrix into a single vector, see Henderson and Searle (1979).

For Hessian matrices, there is not much gain from using this vectorized form since the matrix form is already widely analyzed. However for $r > 2$, this vectorized characterization, which maintains all derivatives as vectors, has contributed to recent advances in multivariate analysis which have been long hindered by the lack of suitable analytical tools. For example, Chacón and Duong (2010, 2011) and Chacón et al. (2011) treated higher derivatives involved in multivariate non-parametric data smoothing. These authors relied heavily on the derivatives of the Gaussian density function, as defined in terms of the Hermite polynomials.

Let $\phi(\boldsymbol{x}) = (2\pi)^{-d/2} \exp(-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x})$ be the standard d-dimensional Gaussian density and $\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = |\boldsymbol{\Sigma}|^{-1/2} \phi(\boldsymbol{\Sigma}^{-1/2}\boldsymbol{x})$ be the centred Gaussian density with variance $\boldsymbol{\Sigma}$. Holmquist (1996a) showed that the r-th derivative of $\phi_{\boldsymbol{\Sigma}}$ is

$$\mathsf{D}^{\otimes r} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = (-1)^r (\boldsymbol{\Sigma}^{-1})^{\otimes r} \mathcal{H}_r(\boldsymbol{x}; \boldsymbol{\Sigma}) \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}), \tag{1}$$

where the r-th order Hermite polynomial is defined by

$$\mathcal{H}_r(\boldsymbol{x}; \boldsymbol{\Sigma}) = r! \mathcal{S}_{d,r} \sum_{j=0}^{[r/2]} \frac{(-1)^j}{j!(r-2j)!2^j} \{\boldsymbol{x}^{\otimes(r-2j)} \otimes (\operatorname{vec} \boldsymbol{\Sigma})^{\otimes j}\}. \tag{2}$$

Here $\mathcal{S}_{d,r}$ is the $d^r \times d^r$ symmetrizer matrix defined as

$$\mathcal{S}_{d,r} = \frac{1}{r!} \sum_{i_1, i_2, \ldots, i_r = 1}^{d} \sum_{\sigma \in \mathcal{P}_r} \bigotimes_{\ell=1}^{r} \boldsymbol{e}_{i_\ell} \boldsymbol{e}_{i_{\sigma(\ell)}}^T, \tag{3}$$

with $\mathcal{P}_r$ standing for the group of permutations of order r and $\boldsymbol{e}_i$ for the ith column of $\mathbf{I}_d$, the identity matrix of order d. We also have that $\mathcal{S}_{d,0} = 1$ and $\mathcal{S}_{d,1} = \mathbf{I}_d$. This definition is highly abstract so we take a concrete example to demonstrate the action of this symmetrizer matrix on a threefold product, i.e. $\mathcal{S}_{d,3}(\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3) = \frac{1}{6}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3 + \boldsymbol{x}_1 \otimes \boldsymbol{x}_3 \otimes \boldsymbol{x}_2 + \boldsymbol{x}_2 \otimes \boldsymbol{x}_1 \otimes \boldsymbol{x}_3 + \boldsymbol{x}_2 \otimes \boldsymbol{x}_3 \otimes \boldsymbol{x}_1 + \boldsymbol{x}_3 \otimes \boldsymbol{x}_1 \otimes \boldsymbol{x}_2 + \boldsymbol{x}_3 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_1]$. In general, the symmetrizer matrix $\mathcal{S}_{d,r}$ maps the product $\bigotimes_{i=1}^r \boldsymbol{x}_i$ to an equally weighted linear combination of products of all possible permutations of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_r$.

The goal of this paper is to investigate efficient ways to compute the r-th derivative $\mathsf{D}^{\otimes r} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$ of the multivariate Gaussian density function, and their applications to several statistical problems.

3 Recursive computation of the symmetrizer matrix

Surely the most prohibitive element in the computation of the r-th derivative of the d-variate Gaussian density is the

symmetrizer matrix $\boldsymbol{\mathcal{S}}_{d,r}$. It is a huge matrix, even for low values of d and r (for instance, $\boldsymbol{\mathcal{S}}_{4,8}$ is a matrix of order $65{,}536 \times 65{,}536$) and its definition involves $r!d^r$ summands, hence its direct calculation can be onerous both in memory storage and computational time.

Nevertheless, this symmetrizer matrix has independent interest on its own from an algebraic point of view. This is well certified by the fact that it has been independently discovered many times. To our knowledge, Holmquist (1985) was the first to develop its form as a generalization of Kronecker product permuting matrices. More recently, Schott (2003) and Meijer (2005) found alternative derivations and further interesting properties.

First, to reduce the number of loops in (3) it is useful to consider the conversion to base d. Any number $i \in \{0, 1, \ldots, d^r - 1\}$ can be written in base d as $(a_r \cdots a_2 a_1)_d$, with digits $a_j \in \{0, 1, \ldots, d-1\}$, meaning that $i = \sum_{j=1}^{r} a_j d^{j-1}$. A simple translation yields that the correspondence between the set $\mathcal{PR}_{d,r} = \{(i_1, \ldots, i_r) : i_1, \ldots, i_r \in \{1, \ldots, d\}\}$ of permutations with repetition of d elements, taken r at a time, and the set $\{1, 2, \ldots, d^r\}$, given by $p(i_1, \ldots, i_r) = 1 + \sum_{j=1}^{r}(i_j - 1)d^{j-1}$ is also bijective, hence all r-tuples $(i_1, \ldots, i_r)$ involved in the multi-index for the first summation in (3) can be obtained as $p^{-1}(i)$ as i ranges over $\{1, 2, \ldots, d^r\}$ (see Appendix 1), so that only two loops are needed for the direct computation of the symmetrizer matrix. Moreover, after a careful inspection of the r-fold Kronecker product involved it follows that (3) can be written as

$$\boldsymbol{\mathcal{S}}_{d,r} = \frac{1}{r!} \sum_{i=1}^{d^r} \sum_{\sigma \in \mathcal{P}_r} \mathbf{E}_{i,(p \circ \sigma \circ p^{-1})(i)}, \tag{4}$$

where $\mathbf{E}_{i,j}$ represents the $d^r \times d^r$ matrix having the (i, j)-th element equal to 1 as its only nonzero element. The operator p^{-1} maps an integer i to a unique r-tuple $(i_1, \ldots, i_r)$, the operator σ generates a permutation of a given r-tuple, and the operator p maps an r-tuple to an integer in $\{1, \ldots, d^r\}$. So the composition $(p \circ \sigma \circ p^{-1})$, as i ranges over $\{1, \ldots, d^r\}$ and σ over $\mathcal{P}_r$, generates an equivalent set to the set of permutations defined in (3). Hence, the novel formulation in Eq. (4) is more appropriate for efficient computations.

Even in this simple form, the direct implementation of $\boldsymbol{\mathcal{S}}_{d,r}$ using (4) usually takes a considerable amount of time as d and r increase, due to the large number of terms involved in each of the two loops. A useful way to improve over the direct approach is to use a recursive implementation of $\boldsymbol{\mathcal{S}}_{d,r}$. Thus, the goal of this section is to express the symmetrizer matrix $\boldsymbol{\mathcal{S}}_{d,r+1}$ in terms of the symmetrizer matrix of lower order $\boldsymbol{\mathcal{S}}_{d,r}$.

Let us denote by $\mathcal{M}_{m \times n}$ the set of all $m \times n$ matrices and let $\mathbf{K}_{r,s} \in \mathcal{M}_{rs \times rs}$ be the commutation matrix of order r, s; see Magnus and Neudecker (1979). The commutation matrix allows us to commute the order of the matrices in a Kronecker product, e.g., if $\mathbf{A} \in \mathcal{M}_{m \times n}$ and $\mathbf{B} \in \mathcal{M}_{p \times q}$, then $\mathbf{K}_{p,m}(\mathbf{A} \otimes \mathbf{B})\mathbf{K}_{n,q} = \mathbf{B} \otimes \mathbf{A}$. The relationship between $\boldsymbol{\mathcal{S}}_{d,r+1}$ and $\boldsymbol{\mathcal{S}}_{d,r}$ is given in the next theorem.

Theorem 1 *Consider the matrix* $\mathbf{T}_{d,r} \in \mathcal{M}_{d^r \times d^r}$ *defined by*

$$\mathbf{T}_{d,r} = \frac{1}{r} \sum_{j=1}^{r} (\mathbf{I}_{d^j} \otimes \mathbf{K}_{d^{r-j-1},d})(\mathbf{I}_{d^{j-1}} \otimes \mathbf{K}_{d,d^{r-j}})$$

where, by convention, $\mathbf{K}_{d^{-1},d} = 1 \in \mathbb{R}$. *Then* $\boldsymbol{\mathcal{S}}_{d,r+1} = (\boldsymbol{\mathcal{S}}_{d,r} \otimes \mathbf{I}_d)\mathbf{T}_{d,r+1}$.

From Theorem 1 it follows that, to obtain a recursive formula for $\boldsymbol{\mathcal{S}}_{d,r}$, it suffices to obtain a recursive formula for $\mathbf{T}_{d,r}$. This is provided in the next result.

Theorem 2 *For any* $r \geq 1$ *the relationship between* $\mathbf{T}_{d,r+1}$ *and* $\mathbf{T}_{d,r}$ *is given by*

$$(r+1)\mathbf{T}_{d,r+1} = (\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d})(r\mathbf{T}_{d,r} \otimes \mathbf{I}_d)(\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d}) + \mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d}.$$

Combining Theorems 1 and 2, the proposed recursive algorithm to compute $\boldsymbol{\mathcal{S}}_{d,r}$ reads as follows:

Algorithm 1: Recursive symmetrizer matrix computation

Input : dimension d and order r
Output: Symmetrizer matrix $\boldsymbol{\mathcal{S}}_{d,r}$
1. If $r = 0$ set $\boldsymbol{\mathcal{S}}_{d,r} = 1$
2. If $r = 1$ set $\boldsymbol{\mathcal{S}}_{d,r} = \mathbf{I}_d$
3. If $r \geq 2$ then set $\mathbf{S} = \mathbf{T} = \mathbf{I}_d$ and $\mathbf{A} = \mathbf{K}_{d,d}$
   For i in $2, \ldots, r$:
   Set $\mathbf{T} = \mathbf{A}(\mathbf{T} \otimes \mathbf{I}_d)\mathbf{A} + \mathbf{A}$ and $\mathbf{S} = (\mathbf{S} \otimes \mathbf{I}_d)\mathbf{T}$
   If $i < r$, set $\mathbf{A} = \mathbf{I}_d \otimes \mathbf{A}$
4. Return $\boldsymbol{\mathcal{S}}_{d,r} = \mathbf{S}/r!$

The proofs of all the new results in the paper, including Theorems 1 and 2, will be deferred to Appendix 1. Besides, a detailed comparison of the computation times for the direct approach [based on Eq. (4)] and the new recursive Algorithm 1 is given below in Sect. 7.

4 Recursive computation of the product of the symmetrizer matrix and a vector

Although the computation of symmetrizer matrices has an algebraic interest on its own, recall from the Introduction that the primary motivation for the name of the symmetrizer matrix is its symmetrizing action on a Kronecker product vector. Thus, for a vector $\boldsymbol{v} = (v_1, \ldots, v_{d^r})$, the product $\boldsymbol{\mathcal{S}}_{d,r}\boldsymbol{v}$ deserves to be studied more closely. For example, when the

final goal is to obtain the r-th order Hermite polynomial it may not be strictly necessary to compute $\mathcal{S}_{d,r}$ explicitly. To understand this notice that, from (4), the i-th coordinate of the vector $\boldsymbol{w} = \mathcal{S}_{d,r}\boldsymbol{v}$ is just

$$w_i = \frac{1}{r!}\sum_{\sigma\in\mathcal{P}_r} v_{(p\circ\sigma\circ p^{-1})(i)}. \tag{5}$$

This makes it feasible to obtain $\mathcal{S}_{d,r}\boldsymbol{v}$ for higher values of d and r, in situations where memory limitations do not allow us to compute the whole matrix $\mathcal{S}_{d,r}$.

The recursive approach to compute $\mathcal{S}_{d,r}\boldsymbol{v}$ is based on the following corollary of Theorem 1, in which we show that by induction it is possible to obtain a new representation of the symmetrizer matrix, a factorization with r factors depending only on the $\mathbf{T}_{d,k}$ matrices for $k = 1, \ldots, r$.

Corollary 1 *For any $r = 1, 2, \ldots$, the symmetrizer matrix can be factorized as*

$$\begin{aligned}\mathcal{S}_{d,r} &= \prod_{k=1}^{r}(\mathbf{T}_{d,k}\otimes\mathbf{I}_{d^{r-k}})\\ &= (\mathbf{T}_{d,1}\otimes\mathbf{I}_{d^{r-1}})(\mathbf{T}_{d,2}\otimes\mathbf{I}_{d^{r-2}})\cdots(\mathbf{T}_{d,r-1}\otimes\mathbf{I}_{d})\mathbf{T}_{d,r}.\end{aligned}$$

This factorization can be further simplified by noting that $\mathbf{T}_{d,1} = \mathbf{I}_d$.

Corollary 1 suggests a straightforward recursive scheme provided a simple formula for each of the factors $(\mathbf{T}_{d,k}\otimes\mathbf{I}_{d^{r-k}})\boldsymbol{v}$ is available. We derive such a formula in the next result.

Corollary 2 *Denote by τ_{jk} the transposition that interchanges the j-th and k-th coordinates of an index vector $(i_1, \ldots, i_r)$ with $1 \le i_1, \ldots, i_r \le d$. For any vector $\boldsymbol{v} = (v_1, \ldots, v_{d^r}) \in \mathbb{R}^{d^r}$ and $k = 2, \ldots, r$, it is possible to express $(\mathbf{T}_{d,k}\otimes\mathbf{I}_{d^{r-k}})\boldsymbol{v} = \frac{1}{k}\sum_{j=1}^{k}\boldsymbol{w}_{p\circ\tau_{jk}\circ p^{-1}}$, where $\boldsymbol{w}_{p\circ\tau_{jk}\circ p^{-1}} \in \mathbb{R}^{d^r}$ is the vector whose $(p\circ\tau_{jk}\circ p^{-1})(i)$-th coordinate is v_i.*

From Corollary 2 it follows that, once the set $\mathcal{PR}_{d,r} = \{p^{-1}(i)\colon i = 1, \ldots, d^r\}$ of permutations with repetitions has been obtained (see Appendix 1), the vector $\mathcal{S}_{d,r}\boldsymbol{v}$ can be computed using just two nested loops with a small number of iterations, namely for $k = 2, \ldots, r$ and $j = 1, \ldots, k$. This implementation is described in Algorithm 2. Again, we refer to Sect. 7 for the comparison of the computation times of the direct approach [based on Eq. (5)] and the new recursive Algorithm 2.

Algorithm 2: Recursive computation of $\mathcal{S}_{d,r}\boldsymbol{v}$

Input : dimension d, order r, a vector $\boldsymbol{v} \in \mathbb{R}^{d^r}$
Output: The product $\boldsymbol{w} = \mathcal{S}_{d,r}\boldsymbol{v}$

1. Set $\boldsymbol{w}_{old} = \boldsymbol{v}$
2. If $r \ge 2$ then
   (a) Generate the set $\mathcal{PR}_{d,r}$ as described in Appendix 1
   (b) For k in $2, \ldots, r$:
   Initialize $\boldsymbol{w}_{new} = \mathbf{0} \in \mathbb{R}^{d^r}$
   For j in $1, \ldots, k$:
   Add $\boldsymbol{w}_{old}$ to the coordinates of $\boldsymbol{w}_{new}$ reordered according to $p\circ\tau_{jk}\circ p^{-1}$ (as indicated in Corollary 2)
   Set $\boldsymbol{w}_{old} = \boldsymbol{w}_{new}/k$
3. Return $\boldsymbol{w}_{old}$

5 Recursive computation of all the unique partial derivatives of the multivariate Gaussian density

Employing the vectorization $\mathsf{D}^{\otimes r} f$ to encompass all the r-th order partial derivatives into a single vector is quite useful for a neat theoretical analysis of quantities based on multivariate higher-order derivatives. For instance, from the explicit formula for $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}$ given in terms of the multivariate Hermite polynomials, involving $\mathcal{S}_{d,r}$, Holmquist (1988) was able to derive explicit expressions for the moments and cumulants of arbitrary order of the multivariate normal distribution, whereas all the previous studies only presented tailored formulas for a few particular cases (see Sect. 6.1 below).

However, many of the partial derivative operators in the vector $\mathsf{D}^{\otimes r}$ may appear duplicated, due to Schwarz's theorem on the commutation of higher-order partial derivatives. Thus, it would be desirable in practice to avoid computing these elements more than once. For example, when commutation of partial derivatives of second order is allowed, it suffices to compute the terms $\partial^2/\partial x_i^2$ for $i = 1, \ldots, d$, and just $\partial^2/(\partial x_i \partial x_j)$ for $i < j$, to obtain the whole operator $\mathsf{D}^{\otimes 2}$. It is not necessary to compute the mixed partial derivatives for $i > j$. We will refer to this reduced set of partial derivatives, that unequivocally determine the full derivative vector, as the 'unique partial derivatives'. By this phrase, we mean the set of partial derivatives with unique partial derivative *indices*.

This section makes use of this observation to introduce a different approach, in which a further reduction in storage space and computation time is achieved by computing only the unique partial derivatives of $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}$ and re-distributing them later to form the full vector.

First, notice that each coordinate of the operator $\mathsf{D}^{\otimes r}$ can be written as $\mathsf{D}_{\boldsymbol{i}}$ for some $\boldsymbol{i} \in \mathcal{PR}_{d,r} = \big\{(i_1, \ldots, i_r)\colon i_1, \ldots, i_r \in \{1, \ldots, d\}\big\}$, where

$$\mathsf{D}_{\boldsymbol{i}} = \frac{\partial^r}{\partial x_{i_1}\cdots\partial x_{i_r}},$$

so that the index i_j refers to the coordinate with respect to which the j-th partial derivative is performed, for $1 \le j \le r$. As noted in Sect. 3, the application p gives a one-to-one correspondence between $\mathcal{PR}_{d,r}$ and $\{1, \ldots, d^r\}$ so that it induces a natural ordering $\boldsymbol{i}_1 = p^{-1}(1), \ldots, \boldsymbol{i}_{d^r} = p^{-1}(d^r)$

in $\mathcal{PR}_{d,r}$ (this correspondence is written down explicitly in Appendix 1). It is not difficult to check that, in the formal expression of $\mathsf{D}^{\otimes r}$ as a Kronecker power, its coordinates are arranged precisely in that order, that is,

$$\mathsf{D}^{\otimes r} = (\mathsf{D}_{\boldsymbol{i}_1}, \ldots, \mathsf{D}_{\boldsymbol{i}_{d^r}}).$$

Alternatively, when commutation of partial derivatives is possible, the coordinates of $\mathsf{D}^{\otimes r}$ can also be written as $\mathsf{D}^{\boldsymbol{m}}$ for some $\boldsymbol{m} \in \mathcal{I}_{d,r} = \{(m_1, \ldots, m_d) : 0 \leq m_k \leq r,\ |\boldsymbol{m}| = r\}$, where $|\boldsymbol{m}| = \sum_{k=1}^{d} m_k$ and

$$\mathsf{D}^{\boldsymbol{m}} = \frac{\partial^{|\boldsymbol{m}|}}{\partial x_1^{m_1} \cdots \partial x_d^{m_d}}.$$

Therefore, here the index m_k refers to the number of times that it is partially differentiated with respect to x_k, for $1 \leq k \leq d$.

It is clear that for a given $\boldsymbol{i} \in \mathcal{PR}_{d,r}$ the two definitions agree if m_k is set to be the number of times that the k-th coordinate appears in $\boldsymbol{i}$; that is, $m_k = \sum_{j=1}^{r} I_{\{i_j = k\}}$. But for a given $\boldsymbol{m} \in \mathcal{I}_{d,r}$ there might be many possible multi-indices $\boldsymbol{i} \in \mathcal{PR}_{d,r}$ such that $\mathsf{D}^{\boldsymbol{m}} = \mathsf{D}_{\boldsymbol{i}}$. This is because, provided partial differentiation commutation is possible, the set $\{\mathsf{D}^{\boldsymbol{m}} : \boldsymbol{m} \in \mathcal{I}_{d,r}\}$ contains the unique coordinates of $\mathsf{D}^{\otimes r}$.

Moreover, it is not difficult to show (for instance, by induction on d) that the cardinality of $\mathcal{I}_{d,r}$ is $N_{d,r} = |\mathcal{I}_{d,r}| = \binom{r+d-1}{r}$, which is usually much smaller than d^r. So an efficient way to obtain $\mathsf{D}^{\otimes r}$ is to compute its unique $N_{d,r}$ elements $\{\mathsf{D}^{\boldsymbol{m}} : \boldsymbol{m} \in \mathcal{I}_{d,r}\}$ and then rearrange them to form $\mathsf{D}^{\otimes r}$.

If all the unique partial derivatives $\{\mathsf{D}^{\boldsymbol{m}} : \boldsymbol{m} \in \mathcal{I}_{d,r}\}$ are collected in a vector $\mathfrak{D}^r$ of length $N_{d,r}$, there is also a natural ordering according to which its coordinates should be positioned. This ordering is induced by that of $\mathsf{D}^{\otimes r} = (\mathsf{D}_{\boldsymbol{i}_1}, \ldots, \mathsf{D}_{\boldsymbol{i}_{d^r}})$ in a way such that any $\mathsf{D}^{\boldsymbol{m}}$ can be associated with the first value of $j \in \{1, \ldots, d^r\}$ such that $\mathsf{D}^{\boldsymbol{m}} = \mathsf{D}_{\boldsymbol{i}_j}$. For instance, the first element of $\mathfrak{D}^r$ is necessarily $\mathsf{D}^{(r,0,\ldots,0)} = \mathsf{D}_{(1,1,\ldots,1)} = \mathsf{D}_{\boldsymbol{i}_1}$ and the last element of $\mathfrak{D}^r$ is necessarily $\mathsf{D}^{(0,\ldots,0,r)} = \mathsf{D}_{(d,d,\ldots,d)} = \mathsf{D}_{\boldsymbol{i}_{d^r}}$.

For any $\boldsymbol{m} \in \mathcal{I}_{d,r}$, Erdélyi (1953, Section 12.8) showed that $\mathsf{D}^{\boldsymbol{m}}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$ can also be expressed with the aid of a real-valued Hermite polynomial $\mathcal{H}^{\boldsymbol{m}}$ (remember that the bold font notation $\boldsymbol{\mathcal{H}}_r$ is reserved for the vector-valued Hermite polynomial introduced in (2)) in a way such that

$$\mathsf{D}^{\boldsymbol{m}}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = (-1)^{|\boldsymbol{m}|}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})\mathcal{H}^{\boldsymbol{m}}(\boldsymbol{x}; \boldsymbol{\Sigma}).$$

So if we denote by $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ the vector of length $N_{d,r}$ containing as coordinates all the values $\{\mathcal{H}^{\boldsymbol{m}}(\boldsymbol{x}; \boldsymbol{\Sigma}) : \boldsymbol{m} \in \mathcal{I}_{d,r}\}$, arranged in the same order as the elements of $\mathfrak{D}^r\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$, it is possible to write $\mathfrak{D}^r\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = (-1)^r\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$. Thus, by comparison with the definition of $\boldsymbol{\mathcal{H}}_r$, notice that the vector $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ contains the unique coordinates of $(\boldsymbol{\Sigma}^{-1})^{\otimes r}\boldsymbol{\mathcal{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$.

Savits (2006, Theorem 4.1) showed d recursive formulas that are useful to obtain every coordinate of $\boldsymbol{\mathfrak{H}}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ from some of the elements in $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ and $\boldsymbol{\mathfrak{H}}_{r-1}(\boldsymbol{x}; \boldsymbol{\Sigma})$. Namely, if $\mathbf{V} = \boldsymbol{\Sigma}^{-1} = (v_{ij})_{i,j=1}^{d}$ and $\boldsymbol{z} = \mathbf{V}\boldsymbol{x} = (z_1, \ldots, z_d)$ then, for $j = 1, 2, \ldots, d$, Savits (2006) showed that

$$\mathcal{H}^{\boldsymbol{m}+\boldsymbol{e}_j}(\boldsymbol{x}; \boldsymbol{\Sigma}) = z_j\mathcal{H}^{\boldsymbol{m}}(\boldsymbol{x}; \boldsymbol{\Sigma}) - \sum_{k=1}^{d} v_{jk}m_k\mathcal{H}^{\boldsymbol{m}-\boldsymbol{e}_k}(\boldsymbol{x}; \boldsymbol{\Sigma}), \tag{6}$$

where we follow the convention that $\mathcal{H}^{\boldsymbol{m}-\boldsymbol{e}_k}(\boldsymbol{x}; \boldsymbol{\Sigma}) = 1$ if $m_k = 0$.

Here, an algorithm is proposed to obtain recursively the whole Hermite polynomial vector of unique elements $\boldsymbol{\mathfrak{H}}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ from $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ and $\boldsymbol{\mathfrak{H}}_{r-1}(\boldsymbol{x}; \boldsymbol{\Sigma})$, thus maintaining the analogy with the usual univariate recursive formula. In this vector form, the recursion starts with $\boldsymbol{\mathfrak{H}}_0(\boldsymbol{x}; \boldsymbol{\Sigma}) = 1$ and $\boldsymbol{\mathfrak{H}}_1(\boldsymbol{x}; \boldsymbol{\Sigma}) = \boldsymbol{\Sigma}^{-1}\boldsymbol{x}$.

An obvious difficulty is that Hermite polynomial vectors of different orders have different lengths. Besides, if all the d recursive formulas are applied to each element of $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ then $dN_{d,r}$ elements are obtained and $dN_{d,r} > N_{d,r+1}$ if $r \geq 1$, so necessarily some of the obtained elements would be duplicated. Furthermore, it would be desirable, at each step of the recursion, that the newly obtained Hermite polynomial vector keep the correct order of its coordinates.

A recursive procedure to compute the $N_{d,r+1}$-dimensional vector $\boldsymbol{\mathfrak{H}}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ from the $N_{d,r}$-dimensional vector $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ and the $N_{d,r-1}$-dimensional vector $\boldsymbol{\mathfrak{H}}_{r-1}(\boldsymbol{x}; \boldsymbol{\Sigma})$, using the recursive formulas (6), reads as follows:

1. Using (6) with $j = 1$ it is possible to obtain all the Hermite polynomial values corresponding to $\{\boldsymbol{m} + \boldsymbol{e}_1 : \boldsymbol{m} \in \mathcal{I}_{d,r}\} = \{\mathbf{m} \in \mathcal{I}_{d,r+1} : m_1 \geq 1\}$. There are $N_{d,r}$ of them, which are put in the first $N_{d,r}$ positions of $\boldsymbol{\mathfrak{H}}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$. It remains to compute the Hermite values corresponding to $\{\boldsymbol{m} \in \mathcal{I}_{d,r+1} : m_1 = 0\}$, which can be expressed as $\{(0, m_2, \ldots, m_d) : m_2 + \cdots + m_d = r + 1\}$. There are, therefore, $N_{d-1,r+1}$ of them, which is the remaining number of coordinates of $\boldsymbol{\mathfrak{H}}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ to fill in, since $N_{d,r+1} = N_{d,r} + N_{d-1,r+1}$, according to Pascal's rule.
2. Using (6) with $j = 2$ it is possible to obtain all the Hermite polynomial values corresponding to $\{(0, m_2, \ldots, m_d) : m_2 + \cdots + m_d = r + 1, m_2 \geq 1\}$. Reasoning as in the first step it is clear that there are $N_{d-1,r}$ of them, which are obtained by adding $\boldsymbol{e}_2$ to the multi-indices $\boldsymbol{m} \in \mathcal{I}_{d,r}$ of the form $\boldsymbol{m} = (0, m_2, \ldots, m_d)$. Since, inductively, the first $N_{d,r-1}$ coordinates of the vector $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ correspond to multi-indices $\boldsymbol{m} \in \mathcal{I}_{d,r}$ with $m_1 \geq 1$, formula (6) with $j = 2$ should be applied to the remaining last $N_{d,r} - N_{d,r-1} = N_{d-1,r}$ coordinates of $\boldsymbol{\mathfrak{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ to keep the same coherent ordering in the coordinates of $\boldsymbol{\mathfrak{H}}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$.

Moreover, since formula formula (6) with $j = 2$ is applied to multi-indices $\boldsymbol{m} \in \mathcal{I}_{d,r}$ of the form $\boldsymbol{m} = (0, m_2, \ldots, m_d)$, it can be further simplified to take into account that $m_1 = 0$, yielding

$$\mathcal{H}^{\boldsymbol{m}+\boldsymbol{e}_2}(\boldsymbol{x}; \boldsymbol{\Sigma}) = z_2 \mathcal{H}^{\boldsymbol{m}}(\boldsymbol{x}; \boldsymbol{\Sigma}) - \sum_{k=2}^{d} v_{2k} m_k \mathcal{H}^{\boldsymbol{m}-\boldsymbol{e}_k}(\boldsymbol{x}; \boldsymbol{\Sigma}).$$

It remains to compute the Hermite values corresponding to $\{\boldsymbol{m} \in \mathcal{I}_{d,r+1} : m_1 = m_2 = 0\}$, which can be expressed as $\{(0, 0, m_3, \ldots, m_d) : m_3 + \cdots + m_d = r + 1\}$. There are, therefore, $N_{d-2,r+1}$ of them, which is the remaining number of coordinates of $\mathfrak{H}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ to fill in, because $N_{d,r+1} = N_{d,r} + N_{d-1,r} + N_{d-2,r+1}$, according to Pascal's rule.

3. After the $(d-1)$-th step, the first $\sum_{j=1}^{d-1} N_{d-j+1,r}$ coordinates of $\mathfrak{H}_{r+1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ have been computed, and since $N_{d,r+1} = 1 + \sum_{j=1}^{d-1} N_{d-j+1,r}$ by successive application of Pascal's rule, the only coordinate left is the last one, corresponding to $\boldsymbol{m} = (0, \ldots, 0, r+1) \in \mathcal{I}_{d,r+1}$. To compute it we just apply the iterative formula with $j = d$ to the last element of $\mathfrak{H}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$, which corresponds to $(0, \ldots, 0, r) \in \mathcal{I}_{d,r}$, which in this case simplifies to

$$\mathcal{H}^{(0,\ldots,0,r+1)}(\boldsymbol{x}; \boldsymbol{\Sigma}) = z_d \mathcal{H}^{(0,\ldots,0,r)}(\boldsymbol{x}; \boldsymbol{\Sigma}) - v_{dd} r \mathcal{H}^{(0,\ldots,0,r-1)}(\boldsymbol{x}; \boldsymbol{\Sigma}).$$

The previous steps have been merged into Algorithm 3 to derive a novel recursive procedure to compute $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$.

Algorithm 3: Recursive computation of $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$

Input : vector $\boldsymbol{x} \in \mathbb{R}^d$, $d \times d$ matrix $\boldsymbol{\Sigma}$, order r
Output: The vector $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \in \mathbb{R}^{d^r}$
1. Set $\mathfrak{H}_0(\boldsymbol{x}; \boldsymbol{\Sigma}) = 1$ and $\mathfrak{H}_1(\boldsymbol{x}; \boldsymbol{\Sigma}) = \boldsymbol{\Sigma}^{-1}\boldsymbol{x}$
2. If $r \geq 2$ then, for k in $2, \ldots, r$:
   Proceed as in steps 1–3 in the text to obtain $\mathfrak{H}_k(\boldsymbol{x}; \boldsymbol{\Sigma})$ from $\mathfrak{H}_{k-1}(\boldsymbol{x}; \boldsymbol{\Sigma})$ and $\mathfrak{H}_{k-2}(\boldsymbol{x}; \boldsymbol{\Sigma})$
3. Distribute the elements of $\mathfrak{H}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$ to form $(\boldsymbol{\Sigma}^{-1})^{\otimes r}\boldsymbol{\mathcal{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})$
4. Return $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = (-1)^r(\boldsymbol{\Sigma}^{-1})^{\otimes r}\boldsymbol{\mathcal{H}}_r(\boldsymbol{x}; \boldsymbol{\Sigma})\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$

A natural competitor of this algorithm, also in recursive form, but based on the computation of the whole Hermite vector polynomial and not only its unique coordinates, can be derived from Theorem 7.2 in Holmquist (1996a). From this theorem, it follows that the vectors $\boldsymbol{u}_k = (\boldsymbol{\Sigma}^{-1})^{\otimes k}\boldsymbol{\mathcal{H}}_k(\boldsymbol{x}; \boldsymbol{\Sigma})$ satisfy the recurrence relation

$$\boldsymbol{u}_k = \boldsymbol{\mathcal{S}}_{d,k}\big[(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}) \otimes \boldsymbol{u}_{k-1} - (k-1)\big\{(\operatorname{vec}\boldsymbol{\Sigma}^{-1}) \otimes \boldsymbol{u}_{k-2}\big\}\big]. \tag{7}$$

Therefore, a straightforward recursive implementation of the previous formula, making use of Algorithm 2 to calculate (7), allows us to obtain $\mathsf{D}^{\otimes r}\phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = (-1)^r \boldsymbol{u}_r \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$. The performance of these two recursive algorithms as well as the direct alternative is investigated in Sect. 7 below.

6 Applications to selected statistical problems

The multivariate Gaussian density function plays a key role in many statistical problems. A number of them need not only the function itself, but some of its higher-order derivatives. In this section, we illustrate how the previous methods can be used to deal with some selected situations; the performance of the many possible algorithms arising from the application of the previous recursive techniques to each of these problems is discussed in Sect. 7.

6.1 Moments of Gaussian random variables

Perhaps the most widely studied Gaussian-based scalar functions are the moments of the multivariate normal distribution and the expected values of quadratic forms in normal random variables. Many algorithms have been proposed to compute these, which are too numerous to cite all here. Surely the earliest is Isserlis (1918), but more recent attempts includes Kumar (1973), Magnus (1979), Ghazal (1996), Holmquist (1988, 1996b), Triantafyllopoulos (2003), Kan (2008) and Phillips (2010). As noted before, the advantage of the approach of Holmquist (1988, 1996b) is that it produces concise explicit expressions using the symmetrizer matrix, with its corresponding computational disadvantage. The other references tend to focus on more efficient algorithmic approaches where the underlying structure is obscured, making them less amenable for further mathematical analysis. To this end, we wish to derive algorithms which are both computationally efficient and mathematically tractable.

For $\mathbf{X} \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ a d-variate Gaussian random variable with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$, its raw vector moment of order r is defined as $\boldsymbol{\mu}_r = \mathbb{E}(\mathbf{X}^{\otimes r}) \in \mathbb{R}^{d^r}$. Holmquist (1988) showed that an explicit formula for this vector moment for an arbitrary order r is given by

$$\boldsymbol{\mu}_r = r!\boldsymbol{\mathcal{S}}_{d,r} \sum_{j=0}^{[r/2]} \frac{1}{j!(r-2j)!2^j} \big\{\boldsymbol{\mu}^{\otimes(r-2j)} \otimes (\operatorname{vec}\boldsymbol{\Sigma})^{\otimes j}\big\}. \tag{8}$$

Further, Holmquist (1996a, Equation (9.2)) noted that the resemblance between the previous expression and the definition of the multivariate Hermite polynomial (2) can be expressed as

$$\boldsymbol{\mu}_r = \boldsymbol{\mathcal{H}}_r(\boldsymbol{\mu}; -\boldsymbol{\Sigma}). \tag{9}$$

Although the matrix $\boldsymbol{\Sigma}$ in the definition of the vector Hermite polynomial needs to be positive definite so that all

the formulas have a well-defined probabilistic interpretation, Holmquist (1996a) showed that (9) remains valid even if $-\boldsymbol{\Sigma}$ is negative definite. Therefore, the vector moment $\boldsymbol{\mu}_r$ can be efficiently computed using the algorithms introduced in the previous sections.

Many authors, as for instance Triantafyllopoulos (2003), Kan (2008) or Phillips (2010), focus instead on real-valued moments $\mu_{\boldsymbol{i}} = \mathbb{E}(X_{i_1} \cdots X_{i_r})$, where $\mathbf{X} = (X_1, \ldots, X_d)$ and $\boldsymbol{i} = (i_1, \ldots, i_r) \in \mathcal{PR}_{d,r}$. The vector moment $\boldsymbol{\mu}_r$ contains all these real-valued moments as its coordinates (some of them even duplicated), but the main objection that these authors make about this vector moment formulation is about the difficulties encountered at the time of computing the symmetrizer matrix involved in (8), so they propose different alternatives to compute a single one of these real-valued moments. The approach described above overcomes these difficulties and allows to readily obtain all the real-valued moments at once by computing the whole vector moment.

6.2 Quadratic forms in Gaussian random variables

A closely related problem is that of computing the mixed moment of orders (r, s) of two quadratic forms in normal variables, defined as

$$\nu_{r,s}(\mathbf{A}, \mathbf{B}) \equiv \nu_{r,s}(\mathbf{A}, \mathbf{B}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}[(\mathbf{X}^\top \mathbf{A}\mathbf{X})^r (\mathbf{X}^\top \mathbf{B}\mathbf{X})^s],$$

where $\mathbf{A}, \mathbf{B}$ are both $d \times d$ symmetric matrices and $\mathbf{X} \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Note that by taking $s = 0$ and $\mathbf{B} = \mathbf{I}_d$ (say), the previous functional reduces to the r-th moment of a single quadratic form in normal variables, which will be denoted as $\nu_r(\mathbf{A}) \equiv \nu_r(\mathbf{A}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}[(\mathbf{X}^\top \mathbf{A}\mathbf{X})^r]$.

The connection between these functionals and the vector moments of the multivariate normal distribution was highlighted by Holmquist (1996b), who noted that

$$\nu_{r,s}(\mathbf{A}, \mathbf{B}) = [(\mathrm{vec}^\top \mathbf{A})^{\otimes r} \otimes (\mathrm{vec}^\top \mathbf{B})^{\otimes s}]\boldsymbol{\mu}_{2r+2s} \tag{10}$$

and, consequently, $\nu_r(\mathbf{A}) = (\mathrm{vec}^\top \mathbf{A})^{\otimes r}\boldsymbol{\mu}_{2r}$. This Kronecker product form has the advantage of decoupling the deterministic matrix product $(\mathrm{vec}\,\mathbf{A})^{\otimes r}$ from the raw moment $\boldsymbol{\mu}_{2r}$ of the random vector $\mathbf{X}$. Moreover, Eq. (10) makes it immediate to obtain a general formula for $\nu_{r,s}(\mathbf{A}, \mathbf{B})$ for arbitrary orders (r, s) from (8), as shown in Theorems 2 and 8 of Holmquist (1996b), and, besides, it makes the advantage of using vector moments (as opposite to real-valued moments) more apparent. Furthermore, it also suggests a straightforward way to apply the efficient procedures for the computation of $\boldsymbol{\mu}_{2r+2s}$ in Sect. 6.1 to obtain $\nu_{r,s}(\mathbf{A}, \mathbf{B})$.

However, even if Eq. (10) relates the two types of moments in a simple way, these two moments are quite different in nature. Whereas $\boldsymbol{\mu}_{2r+2s}$ is a high-dimensional vector, $\nu_{r,s}(\mathbf{A}, \mathbf{B})$ is a scalar, so in this case it might be preferable to use an alternative recursive implementation not relying on the computation of such a high-dimensional vector.

The classical alternative approach is based on recursive relation between cumulants and lower order ν functionals. Recall that when the cumulant generating function $\psi(t) = \log \mathbb{E}[\exp\{tY\}]$ of a real random variable Y is r-times differentiable, its r-th cumulant is defined as $\psi^{(r)}(0)$ for $r \geq 1$. Mathai and Provost (1992, Theorem 3.2b.2) asserted that for $r \geq 1$,

$$\nu_r(\mathbf{A}) = \sum_{i=0}^{r-1} \binom{r-1}{i} \kappa_{r-i}(\mathbf{A})\nu_i(\mathbf{A})$$

where $\kappa_r(\mathbf{A})$ is the r-th cumulant of the random variable $\mathbf{X}^\top \mathbf{A}\mathbf{X}$, given by

$$\begin{aligned}\kappa_r(\mathbf{A}) &\equiv \kappa_r(\mathbf{A}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= 2^{r-1}(r-1)!\big[\,\mathrm{tr}\{(\mathbf{A}\boldsymbol{\Sigma})^r\} + r\boldsymbol{\mu}^\top(\mathbf{A}\boldsymbol{\Sigma})^{r-1}\mathbf{A}\boldsymbol{\mu}\big].\end{aligned}$$

The recursion starts with $\nu_0(\mathbf{A}) = 1$.

For the mixed moment $\nu_{r,s}(\mathbf{A}, \mathbf{B})$, Smith (1995, Equation (10)) showed that

$$\nu_{r,s}(\mathbf{A}, \mathbf{B}) = \sum_{i=0}^{r}\sum_{j=0}^{s-1} \binom{r}{i}\binom{s-1}{j} \kappa_{r-i,s-j}(\mathbf{A}, \mathbf{B})\nu_{i,j}(\mathbf{A}, \mathbf{B}) \tag{11}$$

where $\kappa_{r,s}(\mathbf{A}, \mathbf{B})$ is the joint (r, s)-th cumulant of $\mathbf{X}^\top \mathbf{A}\mathbf{X}$ and $\mathbf{X}^\top \mathbf{B}\mathbf{X}$, which is defined as the value at $(0, 0)$ of the (r, s)-th order partial derivative of the joint cumulant generating function $\psi(t_1, t_2) = \log \mathbb{E}[\exp\{t_1 \mathbf{X}^\top \mathbf{A}\mathbf{X} + t_2 \mathbf{X}^\top \mathbf{B}\mathbf{X}\}]$ for $r + s \geq 1$.

Mathai and Provost (1992, Theorem 3.3.4 and Corollary 3.3.1) provided a concise formula for $\kappa_{r,s}(\mathbf{A}, \mathbf{B})$ without an explicit proof. Unfortunately, although their formula is correct for some particular cases, it is wrong in general (see further details in Appendix 1 below). We provide the correct formula for the cumulant $\kappa_{r,s}(\mathbf{A}, \mathbf{B})$ in Theorem 3 below.

Let us denote by $\mathcal{MP}_{r,s}$ the set of permutations of the multiset having r copies of 1 and s copies of 2; that is,

$$\begin{aligned}&\mathcal{MP}_{r,s} \\ &= \big\{\boldsymbol{i} = (i_1, \ldots, i_{r+s}) \in \{1, 2\}^{r+s} : n_1(\boldsymbol{i}) = r,\ n_2(\boldsymbol{i}) = s\big\},\end{aligned}$$

where $n_\ell(\boldsymbol{i})$ denotes the number of times that ℓ appears in $\boldsymbol{i}$, for $\ell = 1, 2$; i.e., $n_\ell(\boldsymbol{i}) = \sum_{k=1}^{r+s} I_{\{i_k = \ell\}}$. Recall that the cardinality of $\mathcal{MP}_{r,s}$ is $(r+s)!/(r!s!)$.

Theorem 3 *For $r + s \geq 1$, the joint cumulant of order (r, s) of $\mathbf{X}^\top \mathbf{A}\mathbf{X}$ and $\mathbf{X}^\top \mathbf{B}\mathbf{X}$ is given by*

$$\begin{aligned}\kappa_{r,s}(\mathbf{A}, \mathbf{B}) = 2^{r+s-1}r!s! \sum_{\boldsymbol{i} \in \mathcal{MP}_{r,s}} \mathrm{tr}\big[\mathbf{F}_{i_1} \cdots \mathbf{F}_{i_{r+s}}\big\{\mathbf{I}_d/(r+s) \\ + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top\big\}\big],\end{aligned}$$

where $\mathbf{F}_1 = \mathbf{A}\boldsymbol{\Sigma}$ and $\mathbf{F}_2 = \mathbf{B}\boldsymbol{\Sigma}$.

The combination of the correct formula for $\kappa_{r,s}(\mathbf{A}, \mathbf{B})$ with (11) results in a straightforward recursive algorithm for the computation of $\nu_{r,s}(\mathbf{A}, \mathbf{B})$, whose performance is reported in Sect. 7.

Upon visual inspection, these ν functionals are composed of various traces of products of $\mathbf{A}$, $\mathbf{B}$ and $\boldsymbol{\Sigma}$, and quadratic forms of these products in $\boldsymbol{\mu}$. Because there exist many results for, say, the differential analysis of these scalar functions, the subsequent differential analysis is no more difficult than the original form in terms of symmetrizer matrices.

6.3 Analysis of Gaussian kernel-based non-parametric data smoothers

A general goal of non-parametric data smoothing is to generate smooth visualizations of discretized data for exploratory data analysis, e.g. see Simonoff (1996) for an overview. Let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ be a random sample drawn from a common density f. The kernel density estimator of $\mathsf{D}^{\otimes r} f$ with Gaussian kernel is $\mathsf{D}^{\otimes r} \hat{f}_{\mathbf{H}}(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} \mathsf{D}^{\otimes r} \phi_{\mathbf{H}}(\boldsymbol{x} - \mathbf{X}_j)$, where $\mathbf{H}$ is a positive definite bandwidth matrix. Hence, all the techniques introduced in the previous sections are quite useful to obtain an efficient implementation of this estimator in practice.

It should be noted that the computation of this kernel density derivative estimator can be expedited using different and complementary approaches to ours if the bandwidth matrix $\mathbf{H}$ is constrained to being a diagonal matrix; e.g. the binned kernel estimators of Wand (1994), and the fast Gauss transform based estimators of Raykar et al. (2010). On the other hand, our goal is to produce efficient algorithms for use with maximally general unconstrained bandwidth matrices.

The crucial factor in the performance of a kernel estimator is the selection of the bandwidth matrix $\mathbf{H}$ of smoothing parameters. The mean integrated squared error (MISE) of the kernel density derivative estimator is defined as

$$\mathrm{MISE}_r(\mathbf{H}) = \mathbb{E} \int \| \mathsf{D}^{\otimes r} \hat{f}_{\mathbf{H}}(\boldsymbol{x}) - \mathsf{D}^{\otimes r} f(\boldsymbol{x}) \|^2 \, d\boldsymbol{x},$$

where $\| \cdot \|$ denotes the usual Euclidean norm. Under suitable regularity conditions, Chacón et al. (2011) showed that as $n \to \infty$ the MISE can be approximated by

$$\begin{aligned} \mathrm{AMISE}_r(\mathbf{H}) &= n^{-1} |\mathbf{H}|^{-1/2} \operatorname{tr} \left((\mathbf{H}^{-1})^{\otimes r} \mathbf{R}(\mathsf{D}^{\otimes r} \phi) \right) \\ &\quad + \tfrac{(-1)^r}{4} [(\operatorname{vec}^\top \mathbf{I}_{d^r}) \otimes (\operatorname{vec}^\top \mathbf{H})^{\otimes 2}] \boldsymbol{\psi}_{2r+4}, \end{aligned}$$

where $\mathbf{R}(\mathbf{g}) = \int \mathbf{g}(\boldsymbol{x}) \mathbf{g}(\boldsymbol{x})^\top \, d\boldsymbol{x}$ for a vector-valued function $\mathbf{g}$, and $\boldsymbol{\psi}_s = \int \mathsf{D}^{\otimes s} f(\boldsymbol{x}) f(\boldsymbol{x}) \, d\boldsymbol{x}$. Thus the minimizer of AMISE_r is a bandwidth matrix with an asymptotically optimal L_2 risk.

The usual approach to select the bandwidth matrix $\mathbf{H}$ from the data is based on first estimating the MISE using the data sample, and then selecting the bandwidth that minimizes the obtained estimate of the MISE. Here, the step regarding the estimation of the MISE typically involves the computation of V-statistics of degree 2 based on higher order derivatives of the Gaussian density function. For instance, the three methods for bandwidth selection proposed in Chacón and Duong (2013) are based, respectively, on the following three estimators of the MISE

$$\begin{aligned} \mathrm{CV}_r(\mathbf{H}) &= (-1)^r \operatorname{vec}^\top \mathbf{I}_{d^r} \left\{ n^{-2} \sum_{i,j=1}^{n} \mathsf{D}^{\otimes 2r} \phi_{2\mathbf{H}}(\mathbf{X}_i - \mathbf{X}_j) \right. \\ &\quad \left. -2[n(n-1)]^{-1} \sum_{i \neq j} \mathsf{D}^{\otimes 2r} \phi_{\mathbf{H}}(\mathbf{X}_i - \mathbf{X}_j) \right\} \\ \mathrm{PI}_r(\mathbf{H}) &= n^{-1} |\mathbf{H}|^{-1/2} \operatorname{tr} \left\{ (\mathbf{H}^{-1})^{\otimes r} \mathbf{R}(\mathsf{D}^{\otimes r} \phi) \right\} \\ &\quad + \tfrac{(-1)^r}{4} [(\operatorname{vec}^\top \mathbf{I}_d)^{\otimes r} \otimes (\operatorname{vec}^\top \mathbf{H})^{\otimes 2}] \hat{\boldsymbol{\psi}}_{2r+4}(\mathbf{G}) \\ \mathrm{SCV}_r(\mathbf{H}) &= n^{-1} |\mathbf{H}|^{-1/2} \operatorname{tr} \left\{ (\mathbf{H}^{-1})^{\otimes r} \mathbf{R}(\mathsf{D}^{\otimes r} \phi) \right\} \\ &\quad + (-1)^r \operatorname{vec}^\top \mathbf{I}_{d^r} n^{-2} \\ &\quad \times \sum_{i,j=1}^{n} \mathsf{D}^{\otimes 2r} \left\{ \phi_{2\mathbf{H}+2\mathbf{G}} - 2\phi_{\mathbf{H}+2\mathbf{G}} + \phi_{2\mathbf{G}} \right\} (\mathbf{X}_i - \mathbf{X}_j) \end{aligned}$$

where $\hat{\boldsymbol{\psi}}_s(\mathbf{G}) = n^{-2} \sum_{i,j=1}^{n} \mathsf{D}^{\otimes s} \phi_{\mathbf{G}}(\mathbf{X}_i - \mathbf{X}_j)$ is a kernel estimator of $\boldsymbol{\psi}_s$ for a given even number s, based on a pilot bandwidth matrix $\mathbf{G}$. These estimators of the MISE are commonly referred to as cross validation, plug-in and smoothed cross validation criteria, respectively.

The zero-th order derivative case poses little problem for computation. However, if higher order derivatives are considered, we quickly run into computational difficulties. Lin and Xi (2010) reduced the computational burden of general U-statistics by aggregating U-statistics of random subsamples. Here, a different approach is taken by seeking computationally efficient forms for the full sample, restricted to V-statistics of degree 2 based on derivatives of the Gaussian density function.

Let us denote $\eta_{r,s}(\boldsymbol{x}; \mathbf{B}, \boldsymbol{\Sigma}) = [(\operatorname{vec}^\top \mathbf{I}_d)^{\otimes r} \otimes (\operatorname{vec}^\top \mathbf{B})^{\otimes s}] \mathsf{D}^{\otimes 2r+2s} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$ for a $d \times d$ symmetric matrix $\mathbf{B}$. Define also $\eta_r(\boldsymbol{x}; \boldsymbol{\Sigma}) \equiv \eta_{r,0}(\boldsymbol{x}; \mathbf{I}_d, \boldsymbol{\Sigma}) = (\operatorname{vec}^\top \mathbf{I}_d)^{\otimes r} \mathsf{D}^{\otimes 2r} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$. It is easy to show that the previous bandwidth selection criteria can be expressed using these functions, so that

$$\begin{aligned} \mathrm{CV}_r(\mathbf{H}) &= (-1)^r \left\{ n^{-2} \sum_{i,j=1}^{n} \eta_r(\mathbf{X}_i - \mathbf{X}_j; 2\mathbf{H}) \right. \\ &\quad \left. -2[n(n-1)]^{-1} \sum_{i \neq j} \eta_r(\mathbf{X}_i - \mathbf{X}_j; \mathbf{H}) \right\}, \\ \mathrm{PI}_r(\mathbf{H}) &= 2^{-(d+r)} \pi^{-d/2} n^{-1} |\mathbf{H}|^{-1/2} \nu_r(\mathbf{H}^{-1}; \mathbf{0}, \mathbf{I}_d) \\ &\quad + \tfrac{(-1)^r}{4} n^{-2} \sum_{i,j=1}^{n} \eta_{r,2}(\mathbf{X}_i - \mathbf{X}_j; \mathbf{H}, \mathbf{G}), \\ \mathrm{SCV}_r(\mathbf{H}) &= 2^{-(d+r)} \pi^{-d/2} n^{-1} |\mathbf{H}|^{-1/2} \nu_r(\mathbf{H}^{-1}; \mathbf{0}, \mathbf{I}_d) \end{aligned}$$

$$+(-1)^r n^{-2} \sum_{i,j=1}^{n} \big\{\eta_r(\mathbf{X}_i - \mathbf{X}_j; 2\mathbf{H} + 2\mathbf{G}) - 2\eta_r(\mathbf{X}_i - \mathbf{X}_j; \mathbf{H} + 2\mathbf{G}) + \eta_r(\mathbf{X}_i - \mathbf{X}_j; 2\mathbf{G})\big\},$$

where it should be noted that the equivalence in the first term of the plug-in and smoothed cross validation criteria follows from Lemma 3.*iv*) in Chacón et al. (2011).

Thus, the key for an efficient implementation of these criteria is to develop a fast recursive algorithm to compute the η functionals. All the developments in the previous sections can be used for this goal by taking into account the following new result.

Theorem 4 *For a fixed $\boldsymbol{x}$, the previous η functionals are related to the ν functionals as follows*

$$\eta_r(\boldsymbol{x}; \boldsymbol{\Sigma}) = \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \nu_r\big(\mathbf{I}_d; \boldsymbol{\Sigma}^{-1}\boldsymbol{x}, -\boldsymbol{\Sigma}^{-1}\big)$$
$$\eta_{r,s}(\boldsymbol{x}; \mathbf{B}, \boldsymbol{\Sigma}) = \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \nu_{r,s}\big(\mathbf{I}_d, \mathbf{B}; \boldsymbol{\Sigma}^{-1}\boldsymbol{x}, -\boldsymbol{\Sigma}^{-1}\big).$$

The recursive formulation allows for a more efficient optimization algorithm to obtain the minimizer of the corresponding bandwidth selection criteria, and these minimizers are commonly used as the basis for data-based optimal bandwidth matrices, whose asymptotic and finite sample properties were studied in Chacón and Duong (2013).

For large n, evaluating the double sum in the previous V-statistics can pose two different, in some sense dual, problems. If we enumerate singly the data difference $\mathbf{X}_i - \mathbf{X}_j$, then this increases the computation time in n^2. If we wish to take advantage of vectorized computations offered in many software packages, then this requires storing an $n^2 \times d$ matrix in memory which is not always feasible. Thus we have to find the right compromise between execution speed and memory usage on commonly available desktops computers.

Following Theorem 4, any V-statistic of the form $Q_r(\boldsymbol{\Sigma}) = n^{-2} \sum_{i,j=1}^{n} \eta_r(\mathbf{X}_i - \mathbf{X}_j; \boldsymbol{\Sigma})$ can be decomposed as a double sum of products of $\nu_r\big(\mathbf{I}_d; \boldsymbol{\Sigma}^{-1}(\mathbf{X}_i - \mathbf{X}_j), -\boldsymbol{\Sigma}^{-1}\big)$ with $\phi_{\boldsymbol{\Sigma}}(\mathbf{X}_i - \mathbf{X}_j)$. The two most computationally intensive steps involve the cumulants

$$\kappa_r\big(\mathbf{I}_d; \boldsymbol{\Sigma}^{-1}(\mathbf{X}_i - \mathbf{X}_j), -\boldsymbol{\Sigma}^{-1}\big) = (-2)^{r-1}(r-1)! \times \big\{-\operatorname{tr}(\boldsymbol{\Sigma}^{-r}) + (\mathbf{X}_i - \mathbf{X}_j)^\top \boldsymbol{\Sigma}^{-r-1}(\mathbf{X}_i - \mathbf{X}_j)\big\}$$

and the normal densities

$$\phi_{\boldsymbol{\Sigma}}(\mathbf{X}_i - \mathbf{X}_j) = (2\pi)^{-d/2} |\boldsymbol{\Sigma}|^{-1/2} \exp\big\{-\tfrac{1}{2}(\mathbf{X}_i - \mathbf{X}_j)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{X}_i - \mathbf{X}_j)\big\}.$$

The time consuming step in common is the double sum of the terms of the form $(\mathbf{X}_i - \mathbf{X}_j)^\top \boldsymbol{\Sigma}^{-\ell}(\mathbf{X}_i - \mathbf{X}_j)$ for some power $\ell \geq 1$ of $\boldsymbol{\Sigma}^{-1}$. If we decouple this term into components

$$(\mathbf{X}_i - \mathbf{X}_j)^\top \boldsymbol{\Sigma}^{-\ell}(\mathbf{X}_i - \mathbf{X}_j) = \mathbf{X}_i^\top \boldsymbol{\Sigma}^{-\ell}\mathbf{X}_i + \mathbf{X}_j^\top \boldsymbol{\Sigma}^{-\ell}\mathbf{X}_j - 2\mathbf{X}_i^\top \boldsymbol{\Sigma}^{-\ell}\mathbf{X}_j, \tag{12}$$

then each of them are efficiently handled by software in terms of execution but with memory requirements only slightly larger than storing the original sample $\mathbf{X}_1, \ldots, \mathbf{X}_n$, since the differences $\mathbf{X}_i - \mathbf{X}_j$, $j = 1, \ldots, n$, are kept in memory for each i singly rather than all for i as we loop over i.

7 Numerical comparisons

The implementation of all the algorithms described in this paper are contained in the `ks` library (Duong 2007) in the `R` statistical programming language (R Core Team 2013), and in a separate, specific script (Online Resource 1) which is also available from the authors' websites. For each scenario, each algorithm was executed 10 times in `R` 3.0.1 under Ubuntu 12.04 LTS 64 bits, installed on a Dell Precision T6700 with 8 Intel Xeon E5-2609@2.40 GHz CPUs and 32 Gb RAM. Since the actual execution times are highly dependent on the computing set-up used, it is more useful to focus on relative execution times to indicate likely performance gains on other computing set-ups.

7.1 Symmetrizer matrix

A carefully designed algorithm for the direct implementation was used so that, in fact, only one of the two loops in (4) is needed, which moreover selects to loop over $i = 1, \ldots, d^r$ if $d^r < r!$ (with $r!$ the cardinality of $\mathcal{P}_r$), and over $\sigma \in \mathcal{P}_r$ otherwise. This direct implementation based on Eq. (4) was compared to the recursive implementation in Algorithm 1, where the ratio of mean direct execution time to the mean recursive execution time are presented in Table 1 for dimension $d = 2, 3, 4$ and order $r = 2, 4, 6, 8$. Due to memory restrictions, the symmetrizer matrix for $d = 4, r = 8$ was not able to be computed.

For $d = 2$, the recursive algorithm seems to be faster from $r = 6$ on, and is already more than 300 times faster for $r = 8$. Thus, for low values of d and large r, the recursive implementation is preferable. But as d increases it is harder

Table 1 Comparison of mean execution times for direct and recursive implementations to compute the symmetrizer matrix $\boldsymbol{\mathcal{S}}_{d,r}$, for dimension $d = 2, 3, 4$ and order $r = 2, 4, 6, 8$

		$r = 2$	$r = 4$	$r = 6$	$r = 8$
$d = 2$	Direct/recursive	0.42	0.75	9.78	386.49
$d = 3$	Direct/recursive	0.20	0.71	0.66	0.52
$d = 4$	Direct/recursive	0.52	0.36	0.04	–

Each row is the ratio of the mean direct time to the mean recursive time

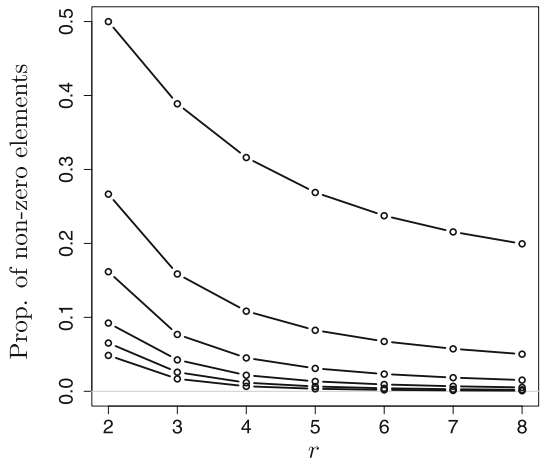

Fig. 1 Proportion of non-zero elements in the lower triangular part of $\mathcal{S}_{d,r}$ as a function of r. From top to bottom the lines correspond to $d = 2, \ldots, 7$

to notice the advantage of the recursive approach, since it is noticeable only for large values of r. Certainly, it must be pointed out that the direct computation of the simplified form (4) makes it quite competitive for low values of d, which are the most commonly used in practice, since handling these huge matrices with the current computational power seems inadvisable for $d \geq 5$ unless r is very low.

In fact, using the direct formula (4) can also be useful to alleviate the problem of the storage space needed, because these sparse matrices have a tiny proportion of non-zero elements, especially for higher values of d. Since the symmetrizer matrices are symmetric (Schott 2003), specifying only its lower triangular part (including the diagonal) suffices to recover the whole matrix. Figure 1 displays the proportion of the $d^r(d^r+1)/2$ entries in the lower triangular part of $\mathcal{S}_{d,r}$ that are not null. Thus, for instance, only 70 elements need to be stored to recover $\mathcal{S}_{7,2}$ (which has 2,401 entries), and only 9,801 elements are needed to recover $\mathcal{S}_{6,4}$ (which has 1,679,616 entries). It remains as an interesting open problem to find an explicit formula for the number of non-zero entries of $\mathcal{S}_{d,r}$.

7.2 Product of a symmetrizer matrix and a vector

A similar experiment was conducted to compare the computation times of the direct approach to compute the product of a symmetrizer matrix and a vector, which is based on Eq. (5), and the recursive approach presented in Algorithm 2. From Table 2, for this problem, the recursive approach proved to be faster than the direct one in all the scenarios. Moreover, the reductions in time achieved by the recursive algorithm can be extremely large for values of d and r commonly encountered in practice. For instance, for $r = 8$ the recursive algorithm produced a result in about 1,000–2,000 times faster. In the previous section, the symmetrizer matrix $\mathcal{S}_{4,8}$ was not able to be computed, whereas the product $\mathcal{S}_{4,8}\boldsymbol{v}$, for $\boldsymbol{v} = (1, 2, \ldots, 4^8)$ posed no memory problems.

Table 2 Comparison of mean execution times for direct and recursive implementations to compute the product of the symmetrizer matrix $\mathcal{S}_{d,r}$ and a d^r-vector, for dimension $d = 2, 3, 4$ and order $r = 2, 4, 6, 8$

		$r=2$	$r=4$	$r=6$	$r=8$
$d=2$	Direct/recursive	4.00	2.83	22.85	1,878.65
$d=3$	Direct/recursive	3.50	2.00	28.03	2,595.11
$d=4$	Direct/recursive	2.00	2.20	21.35	1,150.85

Each row is the ratio of the mean direct time to the mean recursive time

Table 3 Comparison of mean execution times for direct and recursive implementations to compute $\mathsf{D}^{\otimes r}\phi(\cdot)$ the r-th derivative of d-variate standard Gaussian density, for $d = 2, 3, 4$ and order $r = 2, 4, 6, 8, 10$

	$r=2$	$r=4$	$r=6$	$r=8$	$r=10$
$d=2$					
Direct/recursive	1.18	0.68	0.66	0.59	0.66
Direct/unique	1.93	0.76	0.83	1.12	8.00
$d=3$					
Direct/recursive	0.92	0.77	0.65	0.93	0.79
Direct/unique	0.58	0.48	0.62	3.01	7.46
$d=4$					
Direct/recursive	1.00	0.81	0.94	0.96	0.85
Direct/unique	0.48	0.32	0.83	3.93	7.96

In each pair of rows, the upper row is the ratio of mean direct time to the mean time of the first recursive implementation, and the lower row is the ratio of the mean direct time to the mean time of the second recursive implementation where only the unique elements are computed

7.3 Derivatives of a Gaussian density function

We compared the performance of computing the r-th derivative of d-variate standard Gaussian density $\mathsf{D}^{\otimes r}\phi(1, \ldots, 1)$, for dimension $d = 2, 3, 4$ and order $r = 2, 4, 6, 8, 10$. In Table 3, the upper row in each pair of rows compares the direct implementation based on Eqs. (1) and (2), which nevertheless makes use of Algorithm 2 to obtain the multiplication by the symmetrizer matrix, to the first recursive algorithm based on Eq. (7). The lower row in each pair of rows compares the direct implementation to the second recursive algorithm based on Algorithm 3 where only the unique elements are computed. We observed that computing the unique elements of the Hermite vector polynomial eventually becomes faster, as r increases, than the direct and/or first recursive implementations.

Table 4 Comparison of mean execution times for direct and recursive implementations to compute $\boldsymbol{\mu}_r$ the vector moment of a d-variate standard Gaussian random variable, for $d = 2, 3, 4$ and order $r = 2, 4, 6, 8, 10$

	$r=2$	$r=4$	$r=6$	$r=8$	$r=10$
$d=2$					
Direct/recursive	1.79	0.99	0.80	0.82	0.96
Direct/unique	4.67	3.69	1.43	1.20	2.63
$d=3$					
Direct/recursive	3.23	1.37	1.10	1.25	1.13
Direct/unique	6.45	1.00	0.66	2.69	7.14
$d=4$					
Direct/recursive	4.31	1.05	0.76	1.06	1.02
Direct/unique	3.11	0.79	0.57	4.13	7.78

In each pair of rows, the upper row is the ratio of mean direct time to the mean time of the first recursive implementation, and the lower row is the ratio of the mean direct time to the mean time of the second recursive implementation where only the unique elements are computed

7.4 Moments of a Gaussian random variable

We compared the performance of computing the vector r-th moment $\boldsymbol{\mu}_r$ for a standard normal Gaussian $N(0, \mathbf{I}_d)$, for dimension $d = 2, 3, 4$ and order $r = 2, 4, 6, 8, 10$. From Table 4, there does not appear to be a clearly more efficient implementation. In the upper row in each pair of rows, comparing the direct implementation to the first recursive algorithm based on Eqs. (7) and (9), many of these time ratios are around one, indicating a more or less equal computational load. In the lower row in each pair of rows, comparing the direct implementation to the second recursive algorithm where only the unique elements are computed based on Algorithm 3 and Eq. (9), the latter tends to be more efficient than the direct and the first recursive implementation.

7.5 Expected value of quadratic forms in Gaussian random variables

Recall that the expected value of (r, s)-the product of the quadratic form for a d-variate Gaussian random variable $\mathbf{X}$ is $\nu_{r,s}(\mathbf{A}, \mathbf{B}) = \mathbb{E}[(\mathbf{X}^\top \mathbf{A}\mathbf{X})^r (\mathbf{X}^\top \mathbf{B}\mathbf{X})^s]$, and that $\nu_{r,s}$ involves the $(2r+2s)$-th moments of $\mathbf{X}$, so we investigated the performance for dimension $d = 2, 3, 4$ and (r, s) such that $1 \le s \le r, r + s \le 5$, with $\mathbf{A} = \mathrm{diag}(1, \ldots, d)$, $\mathbf{B} = \mathrm{diag}(d, \ldots, 1)$. In the upper row in each group of rows of Table 5, comparing the direct implementation based on Eq. (10) to the first recursive algorithm based on Eqs. (7), (9–10), most of these time ratios are around one, indicating a more or less equal computational load. In the middle row in each group of rows, comparing the direct implementation to the second recursive algorithm based on Algorithm 3 and Equations (9–10), where only the unique elements of the vector moment are computed, most of these time ratios are around one for $r + s < 4$, and greater than one for $r + s \ge 4$. In the lower row in each group of rows, comparing the direct implementation to the third

Table 5 Comparison of mean execution times for direct and recursive implementations to compute $\nu_{r,s}$ the expected value of (r, s)-th product of the quadratic form of a d-variate Gaussian random variable, for $d = 2, 3, 4$ and orders (r, s), $1 \le s \le r, r + s \le 5$

	(r,s)					
	(1, 1)	(2, 1)	(2, 2)	(3, 1)	(3, 2)	(4, 1)
$d=2$						
Direct/recursive	1.36	1.22	1.20	0.90	1.08	1.08
Direct/unique	0.75	0.87	1.18	0.87	2.44	2.37
Direct/cumulant	1.07	1.47	1.64	1.83	3.51	5.33
$d=3$						
Direct/recursive	0.94	1.06	1.25	1.25	1.14	1.14
Direct/unique	0.40	0.56	2.86	2.84	7.76	7.74
Direct/cumulant	1.33	2.70	18.21	24.68	228.98	336.00
$d=4$						
Direct/recursive	1.11	1.05	1.35	1.10	0.89	0.88
Direct/unique	0.27	0.74	4.93	4.63	7.17	7.15
Direct/cumulant	1.33	10.20	290.50	334.77	3,797.22	4,823.30

In each group of rows, the upper row is the ratio of mean direct time to the mean time of the first recursive implementation, the middle row is the ratio of the mean direct time to the mean time of the second recursive implementation where only the unique elements of the vector moment are computed, and the last row is the ratio of the mean direct time to the mean time of the third recursive implementation based on moment-cumulants

Table 6 Comparison of mean execution times for direct and recursive implementations to compute Q_r the Gaussian kernel based V-statistic, for dimension $d = 2, 3, 4$, derivative order $r = 0, 2, 4$, and sample size $n = 100$, 1,000, 10,000

		$n=100$			$n=1{,}000$			$n=10{,}000$		
		$r=0$	$r=2$	$r=4$	$r=0$	$r=2$	$r=4$	$r=0$	$r=2$	$r=4$
$d=2$	Direct/cumulant	0.55	2.93	8.48	6.86	23.97	44.33	4.85	17.47	59.33
$d=3$	Direct/cumulant	0.99	4.04	118.05	7.87	42.51	163.42	7.07	32.17	258.38
$d=4$	Direct/cumulant	1.57	7.06	347.71	9.05	64.52	548.14	6.36	46.83	1,010.99

Each row is the ratio of the mean direct time to the mean recursive time based on moment-cumulants

recursive algorithm based on the moment-cumulant results of Eq. (11) and Theorem 3, the computational speed was multiplied by 10- to 1,000-fold for many cases, as d and/or $(r+s)$ increase. For the (r,s) pairs considered, this third cumulants-based recursive form is generally the most efficient approach, with more and more substantial speed-ups as the dimension and/or the derivative order increase.

7.6 Gaussian kernel based V-statistics

Samples of size $n = 100, 1{,}000, 10{,}000$ were drawn from the d-variate standard Gaussian distribution $N(\mathbf{0}, \mathbf{I}_d)$, for $d = 2, 3, 4$, and from these samples the V-statistic $Q_r(\mathbf{I}_d) = n^{-2}\sum_{i,j=1}^{n} \eta_r(\mathbf{X}_i - \mathbf{X}_j; \mathbf{I}_d)$ was computed for $r = 0, 2, 4$. The direct implementation is based on Eqs. (1) and (2) and the recursive cumlants-based algorithm combining Theorem 4 and Eq. (12). As expected, Table 6 shows that the time savings increase with increasing dimension and increasing derivative order, with 10- to 1,000-fold improvements in most cases.

Acknowledgments We thank two anonymous referees for a careful reading of the paper. This work has been partially supported by grants MTM2010-16660 (both authors) and MTM2010-17366 (first author) from the Spanish Ministerio de Ciencia e Innovación. The second author also received funding from the program "Investissements d' avenir" ANR-10-IAIHU-06

Appendix 1: Proofs

Proofs of the results in Section 3

The key elements to prove Theorem 1 are the following two lemmas.

Lemma 1 *For every $j \in \mathbb{N}_{r+1} := \{1, 2, \ldots, r+1\}$ denote by $\tau_j \in \mathcal{P}_{r+1}$ the permutation defined by $\tau_j(j) = r+1$, $\tau_j(r+1) = j$ and $\tau_j(i) = i$ for $j \neq i \neq r+1$. Then we can express*

$$\mathcal{P}_{r+1} = \{\sigma \circ \tau_j : \sigma \in \mathcal{P}_r,\ j \in \mathbb{N}_{r+1}\}.$$

Proof As any $\sigma \in \mathcal{P}_r$ can be thought as an element of $\mathcal{P}_{r+1}$ by defining $\sigma(r+1) = r+1$, consider the map $\varphi\colon \mathcal{P}_r \times \mathbb{N}_{r+1} \to \mathcal{P}_{r+1}$ given by $\varphi(\sigma, j) = \sigma \circ \tau_j$. We conclude by noting that this map is bijective, with inverse given by $\varphi^{-1}(\tilde{\sigma}) = (\sigma, j)$, where $j = \tilde{\sigma}^{-1}(r+1)$ is such that $\tilde{\sigma}(j) = r+1$ and, for $i \in \mathbb{N}_r$, $\sigma(i) = \tilde{\sigma}(i)$ if $\tilde{\sigma}(i) \neq r+1$ and $\sigma(i) = \tilde{\sigma}(r+1)$ if $\tilde{\sigma}(i) = r+1$. □

Lemma 2 *If $\mathbf{A} \in \mathcal{M}_{m\times n}$, $\mathbf{B} \in \mathcal{M}_{p\times q}$ and $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$, then*

$$\mathbf{A}\otimes\boldsymbol{a}^\top\otimes\mathbf{B}\otimes\boldsymbol{b}^\top = (\mathbf{A}\otimes\boldsymbol{b}^\top\otimes\mathbf{B}\otimes\boldsymbol{a}^\top)\cdot(\mathbf{I}_{dn}\otimes\mathbf{K}_{q,d})\cdot(\mathbf{I}_n\otimes\mathbf{K}_{d,dq}).$$

Proof Use the properties of the commutation matrix to first permute $\boldsymbol{a}^\top \otimes \mathbf{B}$ with $\boldsymbol{b}^\top$, keeping $\mathbf{A}$ in the same place, and then to permute $\boldsymbol{a}^\top$ with $\mathbf{B}$ keeping $\mathbf{A}\otimes\boldsymbol{b}^\top$ in the same place. □

The previous lemmas are helpful to manipulate the original definition of $\boldsymbol{\mathcal{S}}_{d,r}$ and thus obtain the proof of Theorem 1.

Proof of Theorem 1 Note that for any two vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$ we have $\boldsymbol{v}\boldsymbol{w}^\top = \boldsymbol{v}\otimes\boldsymbol{w}^\top$. Then, with the identification $\mathcal{P}_r \subset \mathcal{P}_{r+1}$ and the notation τ_j as in Lemma 1, for any $\sigma \in \mathcal{P}_r$ and $j \in \mathbb{N}_{r+1}$,

$$\begin{aligned}\bigotimes_{\ell=1}^{r+1} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\tau_j(\ell))}}^\top &= \bigotimes_{\ell=1}^{j-1} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \otimes \boldsymbol{e}_{i_j}\boldsymbol{e}_{i_{r+1}}^\top \otimes \bigotimes_{\ell=j+1}^{r} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \otimes \boldsymbol{e}_{i_{r+1}}\boldsymbol{e}_{i_{\sigma(j)}}^\top \\ &= \Big\{\bigotimes_{\ell=1}^{r} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \otimes \boldsymbol{e}_{i_{r+1}}\boldsymbol{e}_{i_{r+1}}^\top\Big\} \cdot(\mathbf{I}_{d^j}\otimes\mathbf{K}_{d^{r-j},d})(\mathbf{I}_{d^{j-1}}\otimes\mathbf{K}_{d,d^{r-j+1}}) \quad (13)\end{aligned}$$

where for the second equality we have applied Lemma 2 with $\boldsymbol{a} = \boldsymbol{e}_{i_{r+1}}$, $\boldsymbol{b} = \boldsymbol{e}_{i_{\sigma(j)}}$,

$$\mathbf{A} = \bigotimes_{\ell=1}^{j-1} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \otimes \boldsymbol{e}_{i_j} \in \mathcal{M}_{d^j,d^{j-1}} \quad \text{and}$$

$$\mathbf{B} = \bigotimes_{\ell=j+1}^{r} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \otimes \boldsymbol{e}_{i_{r+1}} \in \mathcal{M}_{d^{r-j+1},d^{r-j}}.$$

Taking Lemma 1, (13) and the definition of $\mathbf{T}_{d,r+1}$ into account,

$$\begin{aligned}&\boldsymbol{\mathcal{S}}_{d,r+1} \\ &= \frac{1}{(r+1)!}\sum_{i_1,i_2,\ldots,i_{r+1}=1}^{d}\ \sum_{\sigma\in\mathcal{P}_{r+1}} \bigotimes_{\ell=1}^{r+1} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \\ &= \frac{1}{(r+1)!}\sum_{i_1,i_2,\ldots,i_{r+1}=1}^{d}\ \sum_{\sigma\in\mathcal{P}_r}\sum_{j=1}^{r+1} \bigotimes_{\ell=1}^{r+1} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\tau_j(\ell))}}^\top \\ &= \frac{1}{r!}\sum_{i_1,i_2,\ldots,i_{r+1}=1}^{d}\ \sum_{\sigma\in\mathcal{P}_r} \Big\{\bigotimes_{\ell=1}^{r} \boldsymbol{e}_{i_\ell}\boldsymbol{e}_{i_{\sigma(\ell)}}^\top \otimes \boldsymbol{e}_{i_{r+1}}\boldsymbol{e}_{i_{r+1}}^\top\Big\}\mathbf{T}_{d,r+1} \\ &= \Big\{\boldsymbol{\mathcal{S}}_{d,r}\otimes\Big(\textstyle\sum_{i_{r+1}=1}^{d} \boldsymbol{e}_{i_{r+1}}\boldsymbol{e}_{i_{r+1}}^\top\Big)\Big\}\mathbf{T}_{d,r+1} \\ &= (\boldsymbol{\mathcal{S}}_{d,r}\otimes\mathbf{I}_d)\mathbf{T}_{d,r+1},\end{aligned}$$

as $\mathbf{I}_d = \sum_{i=1}^{d} \boldsymbol{e}_i\boldsymbol{e}_i^\top$. □

To obtain a recursive formula for the matrix $\mathbf{T}_{d,r}$ we first need to write the matrices $\mathbf{K}_{d^{p+1},d}$ and $\mathbf{K}_{d,d^{p+1}}$ depending on $\mathbf{K}_{d^p,d}$ and $\mathbf{K}_{d,d^p}$, respectively.

Lemma 3 *For any $p \geq 0$*

$$\begin{aligned}\mathbf{K}_{d^{p+1},d} &= (\mathbf{I}_{d^p}\otimes\mathbf{K}_{d,d})(\mathbf{K}_{d^p,d}\otimes\mathbf{I}_d) \\ &= (\mathbf{I}_d\otimes\mathbf{K}_{d^p,d})(\mathbf{K}_{d,d}\otimes\mathbf{I}_{d^p}) \\ \mathbf{K}_{d,d^{p+1}} &= (\mathbf{K}_{d,d^p}\otimes\mathbf{I}_d)(\mathbf{I}_{d^p}\otimes\mathbf{K}_{d,d}) \\ &= (\mathbf{K}_{d,d}\otimes\mathbf{I}_{d^p})(\mathbf{I}_d\otimes\mathbf{K}_{d,d^p}).\end{aligned}$$

Proof Using part *i*) of Theorem 3.1 in Magnus and Neudecker (1979), we can write

$$\begin{aligned}\mathbf{K}_{d^{p+1},d} &= \sum_{j=1}^{d}(\boldsymbol{e}_j^\top \otimes \mathbf{I}_{d^{p+1}} \otimes \boldsymbol{e}_j) = \sum_{j=1}^{d}(\boldsymbol{e}_j^\top \otimes \mathbf{I}_{d^p} \otimes \mathbf{I}_d \otimes \boldsymbol{e}_j)\\ &= (\mathbf{I}_{d^p} \otimes \mathbf{K}_{d,d}) \sum_{j=1}^{d}(\boldsymbol{e}_j^\top \otimes \mathbf{I}_{d^p} \otimes \boldsymbol{e}_j \otimes \mathbf{I}_d)\\ &= (\mathbf{I}_{d^p} \otimes \mathbf{K}_{d,d})(\mathbf{K}_{d^p,d} \otimes \mathbf{I}_d).\end{aligned}$$

The second equality for $\mathbf{K}_{d^{p+1},d}$ follows similarly and the formulas for $\mathbf{K}_{d,d^{p+1}}$ can be derived from the previous ones by noting that $\mathbf{K}_{d,d^{p+1}} = \mathbf{K}_{d^{p+1},d}^\top$. □

Using the previous lemma we obtain a straightforward proof of Theorem 2.

Proof of Theorem 2 Using Lemma 3 for the first $r-1$ terms in the definition of $\mathbf{T}_{d,r+1}$, and the property that $(\mathbf{AC}) \otimes (\mathbf{BD}) = (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D})$, it follows that

$$\begin{aligned}&(r+1)\mathbf{T}_{d,r+1}\\ &= \sum_{j=1}^{r-1}(\mathbf{I}_{d^j} \otimes \mathbf{K}_{d^{r-j},d})(\mathbf{I}_{d^{j-1}} \otimes \mathbf{K}_{d,d^{r-j+1}})\\ &\quad +(\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d}) + \mathbf{I}_{d^{r+1}}\\ &= \sum_{j=1}^{r-1}\left[\mathbf{I}_{d^j} \otimes \{(\mathbf{I}_{d^{r-j-1}} \otimes \mathbf{K}_{d,d})(\mathbf{K}_{d^{r-j-1},d} \otimes \mathbf{I}_d)\}\right]\\ &\quad \times\left[\mathbf{I}_{d^{j-1}} \otimes \{(\mathbf{K}_{d,d^{r-j}} \otimes \mathbf{I}_d)(\mathbf{I}_{d^{r-j}} \otimes \mathbf{K}_{d,d})\}\right]\\ &\quad +(\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d}) + \mathbf{I}_{d^{r+1}}\\ &= (\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d})\\ &\quad \times\left[\left\{\sum_{j=1}^{r}(\mathbf{I}_{d^j} \otimes \mathbf{K}_{d^{r-j-1},d})(\mathbf{I}_{d^{j-1}} \otimes \mathbf{K}_{d,d^{r-j}})\right\} \otimes \mathbf{I}_d\right]\\ &\quad \times(\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d}) + (\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d})\\ &= (\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d})(r\mathbf{T}_{d,r} \otimes \mathbf{I}_d)(\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d})\\ &\quad +(\mathbf{I}_{d^{r-1}} \otimes \mathbf{K}_{d,d}),\end{aligned}$$

where the third equality makes use of $\mathbf{I}_{d^p} \otimes \mathbf{I}_{d^q} = \mathbf{I}_{d^{p+q}}$. □

Proofs of the results in Section 4

As noted in the text, the proof of Corollary 1 follows by induction on r.

Proof of Corollary 1 For $r = 1$ the formula immediately follows, since $\mathcal{S}_{d,1} = \mathbf{I}_d = \mathbf{T}_{d,1}$. The induction step is easily deduced by using formula $\mathcal{S}_{d,r+1} = (\mathcal{S}_{d,r} \otimes \mathbf{I}_d)\mathbf{T}_{d,r+1}$ from Theorem 1 using the same tools as before, taking into account that $\mathbf{I}_{d^p} \otimes \mathbf{I}_d = \mathbf{I}_{d^{p+1}}$ and that $(\mathbf{AC}) \otimes (\mathbf{BD}) = (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D})$. □

Corollary 2 is deduced from Corollary 1 as follows.

Proof of Corollary 2 Clearly, the Kronecker product $\bigotimes_{\ell=1}^{r}\boldsymbol{e}_{i_\ell}$ of r vectors $\boldsymbol{e}_{i_1}, \ldots, \boldsymbol{e}_{i_r}$ of the canonical basis of $\mathbb{R}^d$ gives the $p(i_1, \ldots, i_r)$-th vector of the canonical basis in $\mathbb{R}^{d^r}$ (i.e., the $p(i_1, \ldots, i_r)$-th column of $\mathbf{I}_{d^r}$). Therefore, any vector $\boldsymbol{v} = (v_1, \ldots, v_{d^r}) \in \mathbb{R}^{d^r}$ can be written as $\boldsymbol{v} = \sum_{i=1}^{d^r} v_i \bigotimes_{\ell=1}^{r}\boldsymbol{e}_{(p^{-1}(i))_\ell}$ and so, by linearity, it suffices to obtain a simple formula for expressions of the type $(\mathbf{T}_{d,k} \otimes \mathbf{I}_{d^{r-k}})(\bigotimes_{\ell=1}^{r}\boldsymbol{e}_{i_\ell})$. Further, since $(\mathbf{T}_{d,k} \otimes \mathbf{I}_{d^{r-k}})(\bigotimes_{\ell=1}^{r}\boldsymbol{e}_{i_\ell}) = \{\mathbf{T}_{d,k}(\bigotimes_{\ell=1}^{k}\boldsymbol{e}_{i_\ell})\} \otimes \bigotimes_{\ell=k+1}^{r}\boldsymbol{e}_{i_\ell}$, it follows that it is enough to provide a simple interpretation for the multiplications $\mathbf{T}_{d,k}(\bigotimes_{\ell=1}^{k}\boldsymbol{e}_{i_\ell})$ for $k = 2, \ldots, r$.

Finally, using the properties of the commutation matrix (Magnus and Neudecker 1979), it can be checked that

$$\mathbf{T}_{d,k}\left(\bigotimes_{\ell=1}^{k}\boldsymbol{e}_{i_\ell}\right) = \frac{1}{k}\sum_{j=1}^{k}\left\{\bigotimes_{\ell=1}^{j-1}\boldsymbol{e}_{i_\ell} \otimes \boldsymbol{e}_{i_k} \otimes \bigotimes_{\ell=j+1}^{k-1}\boldsymbol{e}_{i_\ell} \otimes \boldsymbol{e}_{i_j}\right\} \tag{14}$$

with the convention that $\bigotimes_{\ell=j}^{k}\boldsymbol{e}_{i_\ell} = 1$ if $j > k$. In words, $k\mathbf{T}_{d,k}(\bigotimes_{\ell=1}^{k}\boldsymbol{e}_{i_\ell})$ consists of adding up all the possible k-fold Kronecker products in which the last factor is interchanged with the j-th factor, for $j = 1, 2, \ldots, k$. □

Proofs of the results in Section 6

First, let us point out why the formula for the joint cumulant in Corollary 3.3.1 of Mathai and Provost (1992) is not always correct. Using the notation of Theorem 3 above, their formula reads as follows: for $r \geq 1$, $s \geq 1$,

$$\begin{aligned}\kappa_{r,s}(\mathbf{A}, \mathbf{B}) &= 2^{r+s-1}(r+s-1)!\,\mathrm{tr}\left(\mathbf{F}_1^r\mathbf{F}_2^s\right)\\ &\quad +2^{r+s-1}(r+s-2)!\{r(r-1)\\ &\quad \mathrm{tr}\left(\mathbf{F}_1^{r-1}\mathbf{F}_2^s\mathbf{F}_1\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top\right)\\ &\quad +s(s-1)\,\mathrm{tr}\left(\mathbf{F}_2^{s-1}\mathbf{F}_1^r\mathbf{F}_2\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top\right)\\ &\quad +2rs\,\mathrm{tr}\left(\mathbf{F}_1^r\mathbf{F}_2^s\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top\right)\}.\end{aligned} \tag{15}$$

To further simplify our comparison, consider for example the case $\boldsymbol{\mu} = 0$, and $r = s = 2$, so that (15) simply reads $2^3\,6\,\mathrm{tr}\left(\mathbf{F}_1^2\mathbf{F}_2^2\right)$. Writing down explicitly the six elements in $\mathcal{MP}_{2,2}$ and applying the cyclic property of the trace, the correct form from Theorem 3 has

$$\begin{aligned}&2^3\,2!2!\sum_{i\in\mathcal{MP}_{2,2}}\mathrm{tr}\left(\mathbf{F}_{i_1}\mathbf{F}_{i_2}\mathbf{F}_{i_3}\mathbf{F}_{i_4}\right)/4\\ &= 2^3\left\{4\,\mathrm{tr}\left(\mathbf{F}_1^2\mathbf{F}_2^2\right) + 2\,\mathrm{tr}\left(\mathbf{F}_1\mathbf{F}_2\mathbf{F}_1\mathbf{F}_2\right)\right\}\end{aligned}$$

instead. Both formulas involve 6 traces of matrices, all having two factors $\mathbf{F}_1$ and another two factors $\mathbf{F}_2$. However, despite the aforementioned cyclic property of the trace, it is not true in general that $\mathrm{tr}\left(\mathbf{F}_1\mathbf{F}_2\mathbf{F}_1\mathbf{F}_2\right) = \mathrm{tr}\left(\mathbf{F}_1^2\mathbf{F}_2^2\right)$, and that causes an error in formula (15). A similar argument shows the reason why some of the terms involving $\boldsymbol{\mu}$ in (15) are also wrong.

A sufficient condition for formula (15) to be correct is that $\mathbf{F}_1\mathbf{F}_2 = \mathbf{F}_2\mathbf{F}_1$. If that condition holds, then the correct formula for the joint cumulant further simplifies to

$$\kappa_{r,s}(\mathbf{A},\mathbf{B}) = 2^{r+s-1}(r+s-1)!\big\{\operatorname{tr}\left(\mathbf{F}_1^r\mathbf{F}_2^s\right) + (r+s)\operatorname{tr}\left(\mathbf{F}_1^r\mathbf{F}_2^s\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top\right)\big\}.$$

The proof of Theorem 3 is based on Matrix Calculus. Let us introduce some further notation to simplify the calculations. For $i = 1, 2$, denote

$$\mathbf{C}_i \equiv \mathbf{C}_i(t_1,t_2) = (\mathbf{I}_d - 2t_1\mathbf{F}_1 - 2t_2\mathbf{F}_2)^{-1}\mathbf{F}_i$$

and, similarly, $\mathbf{C}_3 \equiv \mathbf{C}_3(t_1,t_2) = (\mathbf{I}_d - 2t_1\mathbf{F}_1 - 2t_2\mathbf{F}_2)^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top$. Taking into account the formula for the differential of the inverse of a matrix given in Magnus and Neudecker (1999, Chapter 8) notice that the introduced notation allows for simple expressions for the following differentials: for any $i \in \{1,2,3\}$ and $j \in \{1,2\}$, $d\mathbf{C}_i = 2\mathbf{C}_j\mathbf{C}_i dt_j$. In words, differentiating any of these matrix functions with respect to t_j consists on pre-multiplying by $2\mathbf{C}_j$.

More generally, for $i_1,\ldots,i_r \in \{1,2\}$, $j \in \{1,2\}$ and $m \in \{1,2,3\}$ we have

$$\begin{aligned} &d(\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_r}\mathbf{C}_m)\\ &\quad= \{d(\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_r})\}\mathbf{C}_m + \mathbf{C}_{i_1}\cdots\mathbf{C}_{i_r}d\mathbf{C}_m\\ &\quad= 2\left\{\sum_{\ell=1}^{r}\left(\prod_{k=1}^{\ell-1}\mathbf{C}_{i_k}\right)(\mathbf{C}_j\mathbf{C}_{i_\ell})\left(\prod_{k=\ell+1}^{r}\mathbf{C}_{i_k}\right)\mathbf{C}_m\right.\\ &\qquad\left.+\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_r}\mathbf{C}_j\mathbf{C}_m\right\}dt_j\\ &\quad= 2\sum_{\ell=1}^{r+1}\left(\prod_{k=1}^{\ell-1}\mathbf{C}_{i_k}\right)\mathbf{C}_j\left(\prod_{k=\ell}^{r}\mathbf{C}_{i_k}\right)\mathbf{C}_m\,dt_j, \end{aligned} \tag{16}$$

where $\prod_{k=a}^{b}\mathbf{C}_{i_k}$ is to be understood as $\mathbf{I}_d$ if $a > b$.

The key tool for the proof of Theorem 3 is the following lemma, which is indeed valid for any matrix function having the properties of $\mathbf{C}_m$ exhibited above.

Lemma 4 *For any $m \in \{1,2,3\}$, consider the function $w(t_1,t_2) = \operatorname{tr}\mathbf{C}_m$. Then,*

$$\frac{\partial^{r+s}}{\partial t_1^r\partial t_2^s}w(t_1,t_2) = 2^{r+s}\,r!s!\sum_{\boldsymbol{i}\in\mathcal{MP}_{r,s}}\operatorname{tr}\left(\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_{r+s}}\mathbf{C}_m\right).$$

Proof From (16) it easily follows that $d^r\mathbf{C}_m = 2^r r!\,\mathbf{C}_1^r\mathbf{C}_m\,dt_1^r$, so that

$$\frac{\partial^r}{\partial t_1^r}w(t_1,t_2) = 2^r\,r!\operatorname{tr}\left(\mathbf{C}_1^r\mathbf{C}_m\right).$$

Hence, to conclude what we need to show is that, for $s = 0,1,2,\ldots$,

$$\frac{\partial^s}{\partial t_2^s}\operatorname{tr}\left(\mathbf{C}_1^r\mathbf{C}_m\right) = 2^s\,s!\sum_{\boldsymbol{i}\in\mathcal{MP}_{r,s}}\operatorname{tr}\left(\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_{r+s}}\mathbf{C}_m\right) \tag{17}$$

To prove (17) we proceed by induction on s, since the initial step corresponding to $s = 0$ is clear. Assuming that (17) is true for the $(s-1)$-th derivative, the induction step consists of showing that the formula also holds for the s-th derivative; that is,

$$\begin{aligned} &\sum_{\boldsymbol{i}\in\mathcal{MP}_{r,s-1}}\frac{\partial}{\partial t_2}\operatorname{tr}\left(\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_{r+s-1}}\mathbf{C}_m\right)\\ &\quad= 2s\sum_{\boldsymbol{i}\in\mathcal{MP}_{r,s}}\operatorname{tr}\left(\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_{r+s}}\mathbf{C}_m\right). \end{aligned} \tag{18}$$

Taking into account (16), to prove (18) it suffices to show that the set

$$\begin{aligned} \mathcal{A}_{r,s} &= \bigcup_{\ell=1}^{r+s}\{(i_1,\ldots,i_{\ell-1},2,i_\ell,\ldots,i_{r+s-1}) : \boldsymbol{i}\in\mathcal{MP}_{r,s-1}\}\\ &= \{(2,i_1,\ldots,i_{r+s-1}) : \boldsymbol{i}\in\mathcal{MP}_{r,s-1}\}\\ &\quad\cup\{(i_1,2,\ldots,i_{r+s-1}) : \boldsymbol{i}\in\mathcal{MP}_{r,s-1}\}\\ &\quad\cup\cdots\cup\{(i_1,\ldots,i_{r+s-1},2) : \boldsymbol{i}\in\mathcal{MP}_{r,s-1}\} \end{aligned}$$

coincides precisely with the multiset that contains s copies of each of the elements of $\mathcal{MP}_{r,s}$. This can be showed as follows: it is clear that all the elements in $\mathcal{A}_{r,s}$ belong to $\mathcal{MP}_{r,s}$. On the hand, notice that any vector $\boldsymbol{i} = (i_1,\ldots,i_{r+s}) \in \mathcal{MP}_{r,s}$ contains the number 2 in exactly s of its coordinates, which can be distributed along any of the $r+s$ positions. If one of those number 2 coordinates is deleted from $\boldsymbol{i}$, the resulting vector belongs to $\mathcal{MP}_{r,s-1}$, and repeating that process for all the s coordinates with the number 2, then s copies of $\boldsymbol{i}$ are found $\mathcal{A}_{r,s}$. □

Making use of Lemma 4 next we prove Theorem 3.

Proof of Theorem 3 Magnus (1986) showed that the joint cumulant generating function of $\mathbf{X}^\top\mathbf{A}\mathbf{X}$ and $\mathbf{X}^\top\mathbf{B}\mathbf{X}$ can be written as $\psi(t_1,t_2) = u(t_1,t_2) - \frac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + v(t_1,t_2)$, where

$$\begin{aligned} &u(t_1,t_2) = -\tfrac{1}{2}\log|\mathbf{I}_d - 2t_1\mathbf{F}_1 - 2t_2\mathbf{F}_2| \quad \text{and}\\ &v(t_1,t_2) = \tfrac{1}{2}\operatorname{tr}\left\{(\mathbf{I}_d - 2t_1\mathbf{F}_1 - 2t_2\mathbf{F}_2)^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^\top\right\}, \end{aligned}$$

with $\mathbf{F}_1 = \mathbf{A}\boldsymbol{\Sigma}$ and $\mathbf{F}_2 = \mathbf{B}\boldsymbol{\Sigma}$. Since for $r+s \geq 1$ the (r,s)-th joint cumulant is defined as $\kappa_{r,s}(\mathbf{A},\mathbf{B}) = \frac{\partial^{r+s}}{\partial t_1^r\partial t_2^s}\psi(0,0)$, it suffices to show that

$$\begin{aligned} \frac{\partial^{r+s}}{\partial t_1^r\partial t_2^s}\psi(t_1,t_2) &= 2^{r+s-1}r!s!\sum_{\boldsymbol{i}\in\mathcal{MP}_{r,s}}\\ &\operatorname{tr}\left[\mathbf{C}_{i_1}\cdots\mathbf{C}_{i_{r+s}}\left\{\mathbf{I}_d/(r+s) + \mathbf{C}_3\right\}\right]. \end{aligned}$$

With the previous notations, $v(t_1,t_2) = \frac{1}{2}\operatorname{tr}\mathbf{C}_3$, so Lemma 4 immediately yields the desired formula for the second summand.

For the first one, combining the chain rule with the formula for the differential of a determinant given in Magnus and

Neudecker (1999, Chapter 8), it follows that $\frac{\partial}{\partial t_1} u(t_1, t_2) = \operatorname{tr} \mathbf{C}_1$. So, applying Lemma 4 to $\frac{\partial}{\partial t_1} u(t_1, t_2)$, we obtain

$$\frac{\partial^{r+s}}{\partial t_1^r \partial t_2^s} u(t_1, t_2) = 2^{r+s-1} (r-1)! s \times \sum_{\boldsymbol{i} \in \mathcal{MP}_{r-1,s}} \operatorname{tr}\left(\mathbf{C}_{i_1} \cdots \mathbf{C}_{i_{r+s-1}} \mathbf{C}_1\right).$$

By the symmetry in (t_1, t_2) and (r, s) of the preceding argument we come to

$$\begin{aligned}(r+s) \times & \frac{\partial^{r+s}}{\partial t_1^r \partial t_2^s} u(t_1, t_2) \\ &= 2^{r+s-1} r! s! \Big\{ \sum_{\boldsymbol{i} \in \mathcal{MP}_{r-1,s}} \operatorname{tr}\left(\mathbf{C}_{i_1} \cdots \mathbf{C}_{i_{r+s-1}} \mathbf{C}_1\right) \\ &\quad + \sum_{\boldsymbol{i} \in \mathcal{MP}_{r,s-1}} \operatorname{tr}\left(\mathbf{C}_{i_1} \cdots \mathbf{C}_{i_{r+s-1}} \mathbf{C}_2\right) \Big\}.\end{aligned}$$

The proof is finished by noting that, clearly,

$$\begin{aligned}\mathcal{MP}_{r,s} &= \{(i_1, \ldots, i_{r+s-1}, 1) : \boldsymbol{i} \in \mathcal{MP}_{r-1,s}\} \\ &\cup \{(i_1, \ldots, i_{r+s-1}, 2) : \boldsymbol{i} \in \mathcal{MP}_{r,s-1}\}.\end{aligned}$$

□

Although Theorem 4 suffices to obtain a fast recursive implementation of the CV, PI and SCV criteria, here a slightly more general version of this result is shown. Let us denote $\widetilde{\eta}_{r,s}(\boldsymbol{x}; \mathbf{A}, \mathbf{B}, \boldsymbol{\Sigma}) = [(\operatorname{vec}^\top \mathbf{A})^{\otimes r} \otimes (\operatorname{vec}^\top \mathbf{B})^{\otimes s}]\, \mathsf{D}^{\otimes 2r+2s} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$ for $d \times d$ symmetric matrices $\mathbf{A}, \mathbf{B}$ and also $\widetilde{\eta}_r(\boldsymbol{x}; \mathbf{A}, \boldsymbol{\Sigma}) \equiv \widetilde{\eta}_{r,0}(\boldsymbol{x}; \mathbf{A}, \mathbf{I}_d, \boldsymbol{\Sigma}) = (\operatorname{vec}^\top \mathbf{A})^{\otimes r} \mathsf{D}^{\otimes 2r} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x})$. Notice that the η functionals can be seen to be particular cases of the $\widetilde{\eta}$ functionals by setting $\mathbf{A} = \mathbf{I}_d$.

Theorem 5 *For a fixed $\boldsymbol{x}$, the previous $\widetilde{\eta}$ functionals are related to the ν functionals as follows*

$$\begin{aligned}\widetilde{\eta}_r(\boldsymbol{x}; \mathbf{A}, \boldsymbol{\Sigma}) &= \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \nu_r\left(\mathbf{A}; \boldsymbol{\Sigma}^{-1}\boldsymbol{x}, -\boldsymbol{\Sigma}^{-1}\right) \\ \widetilde{\eta}_{r,s}(\boldsymbol{x}; \mathbf{A}, \mathbf{B}, \boldsymbol{\Sigma}) &= \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \nu_{r,s}\left(\mathbf{A}, \mathbf{B}; \boldsymbol{\Sigma}^{-1}\boldsymbol{x}, -\boldsymbol{\Sigma}^{-1}\right).\end{aligned}$$

Proof Notice that (9) entails $\nu_r(\mathbf{A}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (\operatorname{vec}^\top \mathbf{A})^{\otimes r} \boldsymbol{\mathcal{H}}_{2r}(\boldsymbol{\mu}; -\boldsymbol{\Sigma})$. And from Theorem 3.1 in Holmquist (1996a), $(\boldsymbol{\Sigma}^{-1})^{\otimes 2r} \boldsymbol{\mathcal{H}}_{2r}(\boldsymbol{x}; \boldsymbol{\Sigma}) = \boldsymbol{\mathcal{H}}_{2r}(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}; \boldsymbol{\Sigma}^{-1})$. Therefore,

$$\begin{aligned}\widetilde{\eta}_r(\boldsymbol{x}; \mathbf{A}, \boldsymbol{\Sigma}) &= (\operatorname{vec}^\top \mathbf{A})^{\otimes r} \mathsf{D}^{\otimes 2r} \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \\ &= \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) (\operatorname{vec}^\top \mathbf{A})^{\otimes r} (\boldsymbol{\Sigma}^{-1})^{\otimes 2r} \boldsymbol{\mathcal{H}}_{2r}(\boldsymbol{x}; \boldsymbol{\Sigma}) \\ &= \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) (\operatorname{vec}^\top \mathbf{A})^{\otimes r} \boldsymbol{\mathcal{H}}_{2r}(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}; \boldsymbol{\Sigma}^{-1}) \\ &= \phi_{\boldsymbol{\Sigma}}(\boldsymbol{x}) \nu_r\left(\mathbf{A}; \boldsymbol{\Sigma}^{-1}\boldsymbol{x}, -\boldsymbol{\Sigma}^{-1}\right),\end{aligned}$$

as desired. The proof for $\widetilde{\eta}_{r,s}$ follows analogously. □

Appendix 2: Generation of all the permutations with repetitions

A preliminary step to the methods described in Sects. 3, 4 and 5 involves generating the set of all the permutations with repetitions $\mathcal{PR}_{d,r}$. This set can be portrayed as a matrix $\mathbf{P}$ of order $d^r \times r$, whose (i, j)-th entry represents the j-th coordinate of the i-th permutation in $\mathcal{PR}_{d,r}$.

Moreover, in view of Sect. 5 it seems convenient to keep the natural order of these permutations induced by the formulation $\mathcal{PR}_{d,r} = \{p^{-1}(i) : i = 1, \ldots, d^r\}$. Hence, in our construction the vector $p^{-1}(i)$ will constitute the i-th row of $\mathbf{P}$.

Let $\lfloor x \rfloor$ denote the integer part of a real number x, that is, the largest integer not greater than x. Then, if $i = p(i_1, \ldots, i_r) = 1 + \sum_{j=1}^r (i_j - 1) d^{j-1}$ with $i_1, \ldots, i_r \in \{1, \ldots, d\}$, it is not hard to show that $\lfloor (i-1)/d^{k-1} \rfloor = \sum_{j=k}^r (i_j - 1) d^{j-k}$ for $k = 1, \ldots, r$, so that the j-th coordinate of the vector $\boldsymbol{i} = (i_1, \ldots, i_r) = p^{-1}(i)$ can be expressed as $i_j = \lfloor (i-1)/d^{j-1} \rfloor - d \lfloor (i-1)/d^j \rfloor + 1$.

Thus, assuming there is a `floor()` function available that can be applied in an element-wise form to a matrix and returns the integer part of each of its entries, the set $\mathcal{PR}_{d,r}$ is efficiently obtained as the matrix $\mathbf{P} = \texttt{floor}(\mathbf{Q}_{-(r+1)}) - d \cdot \texttt{floor}(\mathbf{Q}_{-1}) + 1$, where $\mathbf{Q}$ is a $d^r \times (r+1)$ matrix whose (i, j)-th entry is $(i-1)/d^{j-1}$ for $i = 1, \ldots, d^r$ and $j = 1, \ldots, r+1$, and $\mathbf{Q}_{-k}$ refers to the sub-matrix obtained from $\mathbf{Q}$ by deleting the k-th column.

References

Chacón, J.E., Duong, T.: Multivariate plug-in bandwidth selection with unconstrained pilot bandwidth matrices. Test **19**, 375–398 (2010)

Chacón, J.E., Duong, T.: Unconstrained pilot selectors for smoothed cross validation. Aust. N. Z. J. Stat. **53**, 331–335 (2011)

Chacón, J.E., Duong, T.: Bandwidth selection for multivariate density derivative estimation, with applications to clustering and bump hunting. Electron. J. Stat. **7**, 499–532 (2013)

Chacón, J.E., Duong, T., Wand, M.P.: Asymptotics for general multivariate kernel density derivative estimators. Stat. Sinica **21**, 807–840 (2011)

Duong, T.: ks: Kernel density estimation and kernel discriminant analysis for multivariate data. J. Stat. Softw. **21**(7), 1–16 (2007)

Erdélyi, A.: Higher Transcendental Functions, vol. 2. McGraw-Hill, New York (1953)

Ghazal, G.A.: Recurrence formula for expectations of products of quadratic forms. Stat. Probab. Lett. **27**, 101–109 (1996)

Henderson, H.V., Searle, S.R.: Vec and vech operators for matrices, with some uses in Jacobians and multivariate statistics. Can. J. Stat. **7**, 65–81 (1979)

Holmquist, B.: The direct product permuting matrices. Linear Multilinear Algeb. **17**, 117–141 (1985)

Holmquist, B.: Moments and cumulants of the multivariate normal distribution. Stoch. Anal. Appl. **6**, 273–278 (1988)

Holmquist, B.: The d-variate vector Hermite polynomial of order k. Linear Algeb. Appl. **237**(238), 155–190 (1996a)

Holmquist, B.: Expectations of products of quadratic forms in normal variables. Stoch. Anal. Appl. **14**, 149–164 (1996b)

Isserlis, L.: On a formula for the product–moment coefficient of any order of a normal frequency distribution in any number of variables. Biometrika **12**, 134–139 (1918)

Kan, R.: From moments of sum to moments of product. J. Multivar. Anal. **99**, 542–554 (2008)

Kumar, A.: Expectation of product of quadratic forms. Sankhyā Ser. B **35**, 359–362 (1973)
Lin, N., Xi, R.: Fast surrogates of U-statistics. Comput. Stat. Data Anal. **54**, 16–24 (2010)
Magnus, J.R.: The expectation of products of quadratic forms in normal variables: the practice. Stat. Neerl. **33**, 131–136 (1979)
Magnus, J.R.: The exact moments of a ratio of quadratic forms in normal variables. Ann. Econ. Stat. **4**, 95–109 (1986)
Magnus, J.R., Neudecker, H.: The commutation matrix: some properties and applications. Ann. Stat. **7**, 381–394 (1979)
Magnus, J.R., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometrics, Revised Edition. Wiley, Chichester (1999)
Mathai, A.M., Provost, S.B.: Quadratic Forms in Random Variables: Theory and Applications. Marcel Dekker, New York (1992)
Meijer, E.: Matrix algebra for higher order moments. Linear Algeb. Appl. **410**, 112–134 (2005)
Phillips, K.: R functions to symbolically compute the central moments of the multivariate normal distribution. J. Stat. Softw. **33** Code Snippet 1 (2010)
R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna (2013)
Raykar, V.C., Duraiswami, R., Zhao, L.H.: Fast computation of kernel estimators. J. Comput. Graph. Stat. **19**, 205–220 (2010)
Savits, T.H.: Some statistical applications of Faa di Bruno. J. Multivar. Anal. **97**, 2131–2140 (2006)
Schott, J.R.: Kronecker product permutation matrices and their application to moment matrices of the normal distribution. J. Multivar. Anal. **87**, 177–190 (2003)
Simonoff, J.S.: Smoothing Methods in Statistics. Springer, Berlin (1996)
Smith, P.J.: A recursive formulation of the old problem of obtaining moments from cumulants and vice versa. Am. Stat. **49**, 217–218 (1995)
Triantafyllopoulos, K.: On the central moments of the multidimensional Gaussian distribution. Math. Sci. **28**, 125–128 (2003)
Wand, M.P.: Fast computation of multivariate kernel estimators. J. Comput. Graph. Stat. **3**, 433–445 (1994)