Statistics and Computing (2025) 35:173
https://doi.org/10.1007/s11222-025-10700-z

ORIGINAL PAPER

Exact computation of angular halfspace depth

Rainer Dyckerhoff[1] · **Stanislav Nagy**[2]

Received: 1 February 2025 / Accepted: 27 July 2025 / Published online: 18 August 2025

Abstract
The angular halfspace depth (ahD) was, already in 1987, the first depth function proposed for the nonparametric analysis of directional data. Mainly due to its presumed high computational cost and lack of efficient computational algorithms, it was never widely used in directional data analysis. We address the problem of the exact computation of ahD in any dimension d. We proceed in two steps: (i) We express ahD as a generalized (Euclidean) halfspace depth in dimension $d-1$, using a projection approach. That allows us to develop fast exact computational algorithms for ahD in dimensions $d = 1, 2, 3$. (ii) In spaces of dimension 3]]d 3 we design an inductive procedure that reduces the dimensionality d in the computation of ahD, until the algorithms for $d \leq 3$ can be used. Using our advances we develop a family of powerful algorithms for the computation of ahD in any dimension d. Our procedures are implemented efficiently in C++ with an interface in R. A detailed analysis of the complexity of the novel algorithms is performed. Surprisingly, we show that computing ahD of multiple points with respect to the same dataset is substantially faster than the same task for the classical (Euclidean) halfspace depth.

Keywords Angular halfspace depth · Exact computation · Depth · Directional data analysis

Mathematics Subject Classification 65C60 · 62G35 · 62H11

1 Angular halfspace depth: Nonparametrics for directional data

Directional data analysis deals with the statistics of datasets that naturally live on the surface of the unit sphere $\mathbb{S}^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| = 1\}, d \geq 1$. Directional data have their own peculiarities; the unit sphere is a compact non-linear manifold whose intrinsic structure must be respected when conducting the analysis. Therefore, many statistics techniques known for $\mathbb{R}^d$-valued (multivariate) data are not applicable to directional data, or lose meaning if used blindly.

Many concepts of parametric statistics have been successfully extended to directional data over the last decades (Mardia 1972; Watson 1983; Mardia and Jupp 2000; Ley and Verdebout 2017). Like for multivariate data, nonparametric directional data analysis is however little explored, with one of the few well-established concepts being the statistical depth functions. Both for multivariate data and in the directional case, statistical depths were developed to introduce nonparametric statistics elements, such as rankings, orderings, or quantiles, beyond the univariate setup. Given a dataset $\boldsymbol{X}$ of n points in a multivariate space $\mathcal{X}$ (here we take $\mathbb{R}^d$ or $\mathbb{S}^{d-1}$) and $\boldsymbol{z} \in \mathcal{X}$, the depth $D(\boldsymbol{z}|\boldsymbol{X}) \in [0, \infty)$ is a number that intends to quantify how much "centrally positioned" the point $\boldsymbol{z}$ is with respect to (w.r.t.) the dataset $\boldsymbol{X}$. Points $\boldsymbol{z}$ that are located centrally within the main bulk of mass of $\boldsymbol{X}$ obtain higher depth values, while points that do not fit in the overall geometry of $\boldsymbol{X}$ receive low depth. This introduces a center-outwards ordering of the points in $\mathcal{X}$ generated by $\boldsymbol{X}$. By extension, it enables the construction of many nonparametric statistics tools in more general spaces $\mathcal{X}$. Depth functions have found many applications in, e.g., data visualization (Rousseeuw et al. 1999), robust estimation (Donoho and Gasko 1992), classification (Li et al. 2012), or the construction of tests for more complex data (Li and Liu 2004).

Different approaches to how to evaluate depth have been proposed in the literature. The original depth in $\mathcal{X} = \mathbb{R}^d$ is

✉ Rainer Dyckerhoff
rainer.dyckerhoff@statistik.uni-koeln.de

Stanislav Nagy
nagy@karlin.mff.cuni.cz

[1] Institute of Econometrics and Statistics, University of Cologne, Cologne, Germany

[2] Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

due to Tukey (1975) and is frequently called the *halfspace* (or *Tukey*) *depth*. For $\boldsymbol{z} \in \mathbb{R}^d$ and a dataset $\boldsymbol{X}$ composed of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathbb{R}^d$ it is defined as[1]

$$hD(\boldsymbol{z}|\boldsymbol{X}) = \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}} \#\left\{i = 1, \ldots, n\colon \boldsymbol{p}'\boldsymbol{x}_i \geq \boldsymbol{p}'\boldsymbol{z}\right\}. \tag{1}$$

This is the minimum number of data points $\boldsymbol{x}_i \in \boldsymbol{X}$ contained in a closed halfspace $H_{\boldsymbol{z},\boldsymbol{p}} = \left\{\boldsymbol{y} \in \mathbb{R}^d\colon \boldsymbol{p}'\boldsymbol{y} \geq \boldsymbol{p}'\boldsymbol{z}\right\}$ in $\mathbb{R}^d$ whose boundary passes through $\boldsymbol{z}$. The inner unit normal of $H_{\boldsymbol{z},\boldsymbol{p}}$ is $\boldsymbol{p} \in \mathbb{S}^{d-1}$. We minimize over $\boldsymbol{p} \in \mathbb{S}^{d-1}$ in (1) to obtain the final depth of $\boldsymbol{z}$. For an example of a contour plot of the halfspace depth function of a dataset $\boldsymbol{X}$ in $\mathbb{R}^2$, see the left-hand panel of Figure 1. Since the 1980s, many other depth functions have been proposed for $\mathcal{X} = \mathbb{R}^d$ in the literature; we mention the simplicial depth (Liu 1990), the zonoid depth (Koshevoy and Mosler 1997), and the measure transportation depth (Chernozhukov et al. 2017). For excellent accounts on the general topic of depth functions and their applications in $\mathbb{R}^d$, we refer to Liu et al. (1999), Zuo and Serfling (2000), and Mosler and Mozharovskyi (2022).

The first depth applicable to directional data was proposed by Small (1987, Example 2.3.4). It is the *angular halfspace depth*, a modification of the halfspace depth (1) to $\mathcal{X} = \mathbb{S}^{d-1}$. For $\boldsymbol{z} \in \mathbb{S}^{d-1}$ and a dataset $\boldsymbol{X}$ of observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathbb{S}^{d-1}$, it is defined as

$$\begin{aligned} ahD(\boldsymbol{z}|\boldsymbol{X}) &= \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}\colon \boldsymbol{p}'\boldsymbol{z}\geq 0} \#\left\{i\colon \boldsymbol{p}'\boldsymbol{x}_i \geq 0\right\} \\ &= \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}\colon \boldsymbol{z}\in H_{\boldsymbol{0},\boldsymbol{p}}} \#\left\{i\colon \boldsymbol{x}_i \in H_{\boldsymbol{0},\boldsymbol{p}}\right\}. \end{aligned} \tag{2}$$

Similarly to the halfspace depth (1), the angular halfspace depth evaluates the minimum number of data points in a closed halfspace $H_{\boldsymbol{0},\boldsymbol{p}} = \left\{\boldsymbol{y} \in \mathbb{R}^d\colon \boldsymbol{p}'\boldsymbol{y} \geq 0\right\}$ in $\mathbb{R}^d$ containing $\boldsymbol{z}$. This time, however, the origin $\boldsymbol{0}$ of $\mathbb{R}^d$ is supposed to lie on the boundary of $H_{\boldsymbol{0},\boldsymbol{p}}$, and only those halfspaces $H_{\boldsymbol{0},\boldsymbol{p}}$ that contain $\boldsymbol{z}$ are considered in (2). Since the intersection of $H_{\boldsymbol{0},\boldsymbol{p}}$ with the unit sphere $\mathbb{S}^{d-1}$ is a closed hemisphere, the angular halfspace depth of a point $\boldsymbol{z}$ can be interpreted as the minimum number of data points $\boldsymbol{x}_i \in \boldsymbol{X}$ contained in a closed hemisphere containing $\boldsymbol{z}$. An example of a dataset in $\mathbb{S}^2$ with its angular halfspace depth is in the right-hand panel of Figure 1. Soon after the angular halfspace depth was introduced, Liu and Singh (1992, Section 4) considered ahD from the theoretical point of view and listed several of its properties. In Liu and Singh (1992, Section 4), the depth ahD was considered mainly as an approach complementary to the newly proposed directional extension of the simplicial depth. Additional directional depth functions have been considered in the literature in the last decade (Agostinelli and Romanazzi 2013; Ley et al. 2014; Pandolfo et al. 2018a, b; Buttarazzi et al. 2018; Konen and Paindaveine 2023; Hallin et al. 2024). Just as the halfspace depth (1) in $\mathbb{R}^d$, the angular halfspace depth (2) nevertheless remains a prime example of a depth proposed for directional data.

It might come as a surprise that after Liu and Singh (1992), most of the literature on directional depth functions did not consider ahD—two recent exceptions are Nagy et al. (2024) and Nagy and Laketa (2025), where the theoretical properties of ahD were examined. A plain answer to why ahD has received so little attention can be found in Pandolfo et al. (2018b, p. 594):

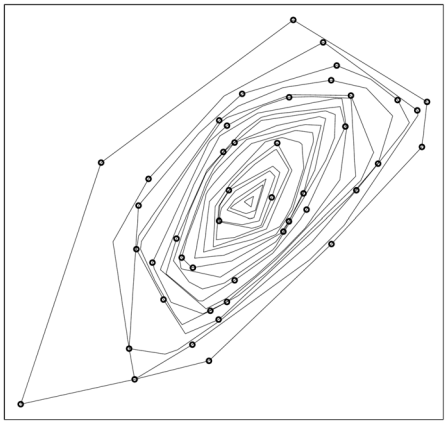
> "*The main drawback of the angular halfspace depth is the computational effort it requires, especially for higher dimensions d*."

Indeed, unlike for the (Euclidean) halfspace depth (1) whose computation is known to be involved, yet already quite well explored (see, e.g., Rousseeuw and Struyf 1998; Miller et al. 2003; Chen et al. 2013; Dyckerhoff and Mozharovskyi 2016; Liu et al. 2019), very little research into the computation of the angular halfspace depth exists in the literature. For example, the only implementation of ahD in R (R Core Team 2024) appears to be the function `sdepth` from the package **depth** (Genest et al. 2019). The function `sdepth` handles data in dimensions $d = 2$ and $d = 3$, but as observed already by, e.g., Pandolfo et al. (2018b), it seems to be rather slow already for $d = 3$ and datasets of size n in the range of hundreds of points. This is illustrated in a simple real data example in Figure 2, where the performance of one of the exact computational algorithms developed in this paper is compared with the function `sdepth` in $\mathbb{S}^2$.

In this paper, we address the problem of exact computation of the angular halfspace depth (2) in arbitrary dimension $d \geq 1$. We do so in two steps:

(i) First, in Section 2, we develop a projection approach allowing us to reduce the problem of computing ahD in $\mathbb{S}^{d-1}$ to computing a variant of the (Euclidean) halfspace depth hD in $\mathbb{R}^{d-1}$. This way, we can use the remarkable advances in the computation of hD also in the directional case. In particular, we design fast exact procedures for direct computation of ahD in dimensions $d = 1, 2, 3$.
(ii) In the second step, in Section 3, we devise a set of recursive procedures that reduce the dimensionality of the setup d. If $d > 3$, we proceed inductively and iteratively

[1] The halfspace depth (and the angular halfspace depth) are generally defined in the context of probability measures instead of just datasets. In that situation, $\boldsymbol{X}$ is represented by an empirical measure P in $\mathbb{R}^d$ (or $\mathbb{S}^{d-1}$) that assigns mass $1/n$ to each data point $\boldsymbol{x}_i$. That results in an additional factor of $1/n$ in the definition of these depths. We omit this scaling factor so that the depths take integer values $0, 1, \ldots, n$ instead of $0/n, 1/n, \ldots, n/n$. This slight modification is, of course, made without loss of generality (w.l.o.g.), and trivially all our results also apply to the standard (angular) halfspace depths.



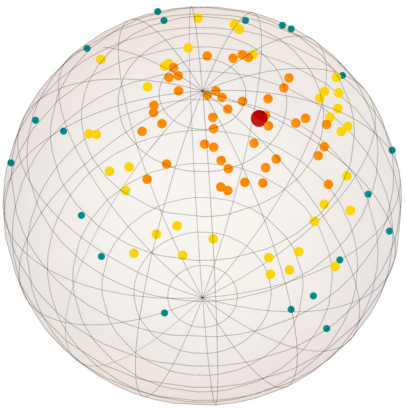


Fig. 1 Left-hand panel: A dataset in $\mathbb{R}^2$ (black points) and the contours of its (Euclidean) halfspace depth hD (polygons with borders in solid line). Right-hand panel: A directional dataset $\boldsymbol{X}$ in $\mathbb{S}^2$ with data points $z \in \mathbb{S}^2$ colored according to their $ahD(z|\boldsymbol{X})$; lower depths in blue, higher depths in orange. The sample point with maximum ahD is displayed as the large red point

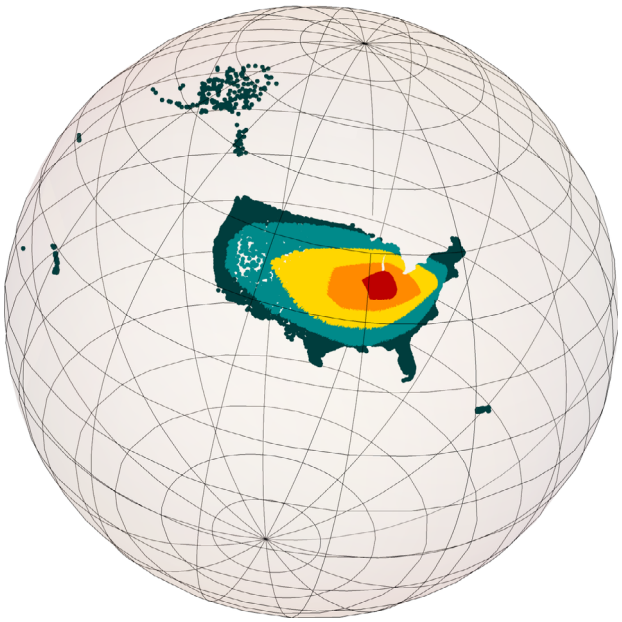


Fig. 2 The exact angular halfspace depth of all $n = 33\,144$ US ZIP Code Tabulation Areas (ZCTA) in 2010. Points with 1% of the highest ahD-values are in red; the deepest ZIP Code is Clinton, IN 47842 in Vermillion County, Indiana, whose depth is $ahD(z|\boldsymbol{X}) = 14\,571$. Computing the exact ahD-values of all $n = 33\,144$ data points using our exact algorithm can be performed simultaneously in approximately 180 seconds. For comparison, the sole competitor of our algorithm, `sdepth` from R package **depth**, takes about 960 seconds to compute the exact ahD of **a single observation** w.r.t. only $m = 1\,000$ data points

lower the dimension of the problem until we reach $d \leq 3$, where the fast procedures from Section 2 are used.

Our programs for computing ahD are implemented efficiently in dimension $d \geq 1$ in C++, with an interface through R. For all our algorithms, we also investigate their computational complexity. We show that the task of computing $ahD(z|\boldsymbol{X})$ in $\mathbb{S}^{d-1}$ is of the same order of complexity as that of computing $hD(z|\boldsymbol{X})$ in $\mathbb{R}^d$. A major difference, however, appears when the depth of k points z_j, $j = 1, \ldots, k$ w.r.t. $\boldsymbol{X}$ is computed simultaneously. Then, our procedures for ahD are surprisingly much faster than what is possible for hD in $\mathbb{R}^d$. In the final Section 4, we conduct a numerical study comparing our algorithms among themselves, and with their sole competitor `sdepth`. The numerical results are impressive—in common situations, the new algorithms can be up to five orders of magnitude faster than the only available competing method, even if implemented in C++.

Notations

The space $\mathbb{R}^d$ is equipped with the Euclidean norm $\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^d x_i^2}$ for $\boldsymbol{x} = (x_1, \ldots, x_d)'$, and a scalar product $\boldsymbol{x}'\boldsymbol{y}$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. We write $\boldsymbol{e}_i$ for the i-th canonical unit vector in $\mathbb{R}^d$; $\boldsymbol{e}_1 = (1, 0, \ldots, 0)'$ etc. The set $H_{\boldsymbol{x},\boldsymbol{p}} = \left\{\boldsymbol{y} \in \mathbb{R}^d : \boldsymbol{y}'\boldsymbol{p} \geq \boldsymbol{x}'\boldsymbol{p}\right\}$ is the (closed) halfspace whose boundary passes through $\boldsymbol{x} \in \mathbb{R}^d$ with inner normal $\boldsymbol{p} \in \mathbb{S}^{d-1}$. For a set $A \subseteq \mathbb{R}^d$, A° is the interior of A, and ∂A is the topological boundary of A. We write span $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ for the linear hull of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. The dimension of an affine space A is denoted by $\dim A$. The orthogonal complement to a linear space A is $A^\perp$. Whenever we write simply $\{i : A(i)\}$, we mean the set of indices $i = 1, \ldots, n$ such that $A(i)$ is true; finally, by $\#A$ we mean the cardinality of the set A.

2 From *ahD* to *hD*: The projection approach

The aim of this section is to design efficient exact computational procedures for the angular halfspace depth in $\mathbb{S}^{d-1}$ in dimensions $d = 1, 2, 3$. This is done by (i) transforming the data from $\mathbb{S}^{d-1}$ into a Euclidean space $\mathbb{R}^{d-1}$ by means of the gnomonic projection of the sphere, and (ii) subsequently expressing ahD as a particular version of the (Euclidean) hD of the projected data. After expounding this projection approach in Section 2.1, in Sections 2.2–2.4 we use this method to design fast algorithms for computing ahD

in dimensions $d = 1, 2, 3$, respectively. We pay close attention to the computational complexity of the new methods. A particular strength of the proposed algorithms is that, in a sharp contrast with the related computation of the conventional hD (Dyckerhoff and Mozharovskyi 2016), the new algorithms are substantially faster in the task of computing the ahD of multiple points w.r.t. the same dataset; this is detailed in Section 2.5. Finally, in Section 2.6 we briefly comment on the possibility of constructing fast approximate algorithms for ahD based on our ideas.

2.1 Angular depth and gnomonic projection

The first observation used throughout this paper is the orthogonal invariance of ahD. The following lemma is proved in, e.g., Small (1987, Example 4.4.4).

Lemma 1 *For a dataset $\boldsymbol{X}$ of n points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathbb{S}^{d-1}$ and an orthogonal matrix $O \in \mathbb{R}^{d\times d}$ let $O\boldsymbol{X}$ be the dataset of n transformed points $O\boldsymbol{x}_1, \ldots, O\boldsymbol{x}_n$ in $\mathbb{S}^{d-1}$. Then we have*

$$ahD(\boldsymbol{z}|\boldsymbol{X}) = ahD(O\boldsymbol{z}|O\boldsymbol{X}) \quad \text{for all } \boldsymbol{z} \in \mathbb{S}^{d-1}.$$

We are given a dataset $\boldsymbol{X}$ and a point $\boldsymbol{z}$ in $\mathbb{S}^{d-1}$. Our task is to compute the angular halfspace depth $ahD(\boldsymbol{z}|\boldsymbol{X})$. Denote by

$$\mathbb{S}_+^{d-1} = \left\{\boldsymbol{y} \in \mathbb{S}^{d-1} \colon \boldsymbol{y}'\boldsymbol{e}_d > 0\right\},$$
$$\mathbb{S}_-^{d-1} = \left\{\boldsymbol{y} \in \mathbb{S}^{d-1} \colon \boldsymbol{y}'\boldsymbol{e}_d < 0\right\},$$
$$\mathbb{S}_0^{d-1} = \left\{\boldsymbol{y} \in \mathbb{S}^{d-1} \colon \boldsymbol{y}'\boldsymbol{e}_d = 0\right\},$$

the (open) positive hemisphere, (open) negative hemisphere, and what we call the equator of the $(d-1)$-sphere, respectively.

Remark 2 To simplify our exposition, we assume, unless stated otherwise, that no points of $\boldsymbol{X}$ lie on the equator $\mathbb{S}_0^{d-1}$, and that $\boldsymbol{z} \in \mathbb{S}_+^{d-1}$. This assumption is made w.l.o.g., by means of Lemma 1.[2]

It will be useful to split the dataset $\boldsymbol{X}$ on $\mathbb{S}^{d-1}$ into two disjoint parts—the points $\boldsymbol{X}_+ = \boldsymbol{X} \cap \mathbb{S}_+^{d-1}$ whose last coordinate is positive, and $\boldsymbol{X}_- = \boldsymbol{X} \cap \mathbb{S}_-^{d-1} = \boldsymbol{X} \setminus \boldsymbol{X}_+$ of points with a negative last coordinate. The principal tool to be used for the computation of $ahD(\boldsymbol{z}|\boldsymbol{X})$ is the *gnomonic projection* of the sphere $\mathbb{S}^{d-1}$ onto the hyperplane

$$\mathcal{G} = \left\{\boldsymbol{y} = (y_1, \ldots, y_d)' \in \mathbb{R}^d \colon y_d = 1\right\} = \partial H_{\boldsymbol{e}_d, \boldsymbol{e}_d}$$

that is tangent to $\mathbb{S}^{d-1}$ at the pole $\boldsymbol{e}_d \in \mathbb{S}^{d-1}$. The gnomonic projection takes points of $\mathbb{S}^{d-1}$ that do not lie on its equator to the unique point of $\mathcal{G}$ on the straight line between $\boldsymbol{x}$ and the origin $\boldsymbol{0}$, see Figure 3. Later in this paper, we will canonically identify the hyperplane $\mathcal{G}$ with $\mathbb{R}^{d-1}$ by dropping the last coordinate 1 of $\boldsymbol{y} \in \mathcal{G}$. In this way, the origin $\boldsymbol{0} \in \mathbb{R}^{d-1}$ is represented by the pole $\boldsymbol{e}_d = \mathcal{G} \cap \mathbb{S}^{d-1}$. The gnomonic projection can then be defined as the mapping

$$\begin{aligned}\xi \colon \mathbb{S}^{d-1} \setminus \mathbb{S}_0^{d-1} &\to \mathcal{G} \colon \boldsymbol{x} = (x_1, \ldots, x_d)' \\ &\mapsto (x_1/x_d, \ldots, x_d/x_d)'. \end{aligned} \tag{3}$$

The gnomonic projection is a double covering of $\mathcal{G}$: once for points from $\mathbb{S}_+^{d-1}$, and once for $\mathbb{S}_-^{d-1}$. Each $\boldsymbol{y} \in \mathcal{G}$ corresponds to exactly one point $\boldsymbol{x}_+ \in \mathbb{S}_+^{d-1}$ such that $\xi(\boldsymbol{x}_+) = \boldsymbol{y}$, and one point $\boldsymbol{x}_- \in \mathbb{S}_-^{d-1}$ with $\xi(\boldsymbol{x}_-) = \boldsymbol{y}$.

Denote by $\xi(\boldsymbol{X}_+)$ and $\xi(\boldsymbol{X}_-)$ the images of the datasets $\boldsymbol{X}_+$ and $\boldsymbol{X}_-$ under the mapping ξ, respectively. That is, $\xi(\boldsymbol{X}_+)$ consists of those points $\xi(\boldsymbol{x}_i) \in \mathcal{G}$ such that $\boldsymbol{x}_i \in \boldsymbol{X}_+$, and analogously for $\xi(\boldsymbol{X}_-)$. We consider $\xi(\boldsymbol{X}_+)$ and $\xi(\boldsymbol{X}_-)$ patched together, and represent it by a signed empirical measure[3] $P_\pm$ in the hyperplane $\mathcal{G}$. The signed measure $P_\pm$ is defined for each Borel set $B \subseteq \mathcal{G}$ by

$$\begin{aligned}P_\pm(B) = &\#\left\{i \colon \boldsymbol{x}_i \in \mathbb{S}_+^{d-1} \text{ and } \xi(\boldsymbol{x}_i) \in B\right\} \\ &-\#\left\{i \colon \boldsymbol{x}_i \in \mathbb{S}_-^{d-1} \text{ and } \xi(\boldsymbol{x}_i) \in B\right\}. \end{aligned} \tag{4}$$

Certainly, $P_\pm$ takes only integer values in the range from $-n$ (if all points from $\boldsymbol{X}$ lie in $\mathbb{S}_-^{d-1}$) to n (if all data points lie in $\mathbb{S}_+^{d-1}$). The signed measure $P_\pm$ puts mass only in the gnomonic projections of all points from $\boldsymbol{X}$, with positive unit mass for each data point mapped from $\mathbb{S}_+^{d-1}$, and negative unit mass to points mapped from $\mathbb{S}_-^{d-1}$. Since we are dealing with datasets, it is, of course, possible that a single point $\boldsymbol{y} \in \mathcal{G}$ corresponds to several pre-image points $\boldsymbol{x}_i$ from different hemispheres, depending on whether $\boldsymbol{x}_i$ came from $\mathbb{S}_+^{d-1}$ or $\mathbb{S}_-^{d-1}$. The $P_\pm$-mass of $\boldsymbol{y}$ then equals

$$P_\pm(\{\boldsymbol{y}\}) = \#\left\{i \colon \xi(\boldsymbol{x}_i) = \boldsymbol{y} \text{ and } \boldsymbol{x}_i \in \mathbb{S}_+^{d-1}\right\}$$

[2] Indeed, if our assumption is not true, we transform all the data $\boldsymbol{X}$ and the point $\boldsymbol{z}$ by a randomly chosen orthogonal transformation O. To be more specific, we sample a vector $\boldsymbol{u}$ from the uniform distribution on $\mathbb{S}^{d-1}$ and apply the Householder transformation (see Golub et al. 2013, Chap. 5) O that maps $\boldsymbol{u}$ to $\boldsymbol{e}_d$. With probability one, this yields a configuration where neither $O\boldsymbol{z}$ nor any data points $O\boldsymbol{X}$ are on the equator. If $O\boldsymbol{z} \in \mathbb{S}_-^{d-1}$, we apply another orthogonal transform $\boldsymbol{y} \mapsto -\boldsymbol{y}$ to flip all points around the origin. Thanks to Lemma 1, $ahD(\boldsymbol{z}|\boldsymbol{X}) = ahD(O\boldsymbol{z}|O\boldsymbol{X}) = ahD(-O\boldsymbol{z}|-O\boldsymbol{X})$, while in one of these setups, our assumption is valid.

[3] Recall that a *signed measure* is a generalized measure that can take both positive and negative values (Dudley 2002, Section 5.6). We say that a measure is *empirical* if it can be written as a sum of finitely many Dirac measures (unit point masses) in $\mathbb{R}^d$. A signed measure $P_\pm$ is *empirical* if there exist empirical measures P_+ and P_- such that $P_\pm = P_+ - P_-$.

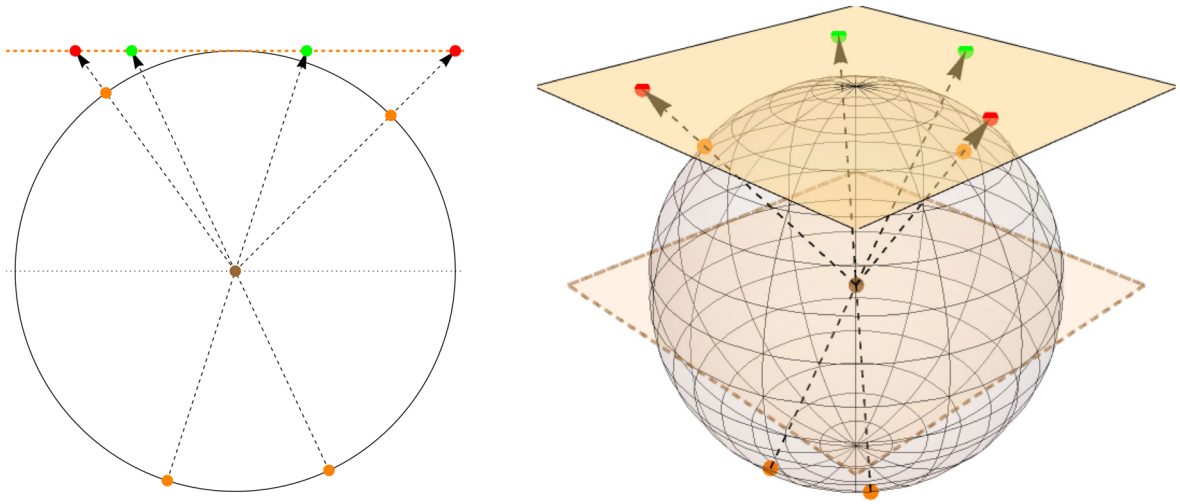


Fig. 3 The gnomonic projection ξ that takes $\mathbb{S}^{d-1}\setminus\mathbb{S}_0^{d-1}$ to $\mathcal{G}$ for $d=2$ (left-hand panel) and $d=3$ (right-hand panel). In both panels we have $n=4$ data points $\boldsymbol{x}_i$ (orange points on $\mathbb{S}^{d-1}$) and their images $\xi(\boldsymbol{x}_i)$ on the dashed line $\mathcal{G}$ in the left-hand panel ($d=2$) and the beige plane $\mathcal{G}$ in the right-hand panel ($d=3$). The images from $\mathbb{S}_+^{d-1}$ with positive $P_\pm$-mass are plotted in red, while the images from $\mathbb{S}_-^{d-1}$ with negative $P_\pm$-mass are displayed in green

$$-\#\left\{i:\xi(\boldsymbol{x}_i)=\boldsymbol{y}\text{ and }\boldsymbol{x}_i\in\mathbb{S}_-^{d-1}\right\}. \tag{5}$$

The signed empirical measure $P_\pm$ on $\mathbb{R}^{d-1}$ can be represented as a list of pairs $\left(\boldsymbol{y}_i,s_i\right)$, $i=1,\ldots,m$, where $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_m$ for $m\le n$ is the set of all distinct images of points from $\boldsymbol{X}$ via (3) in $\mathbb{R}^{d-1}$, and s_i, the *sign* of $\boldsymbol{y}_i$, is the $P_\pm$-mass of $\boldsymbol{y}_i$ given by (5).

For signed empirical measures, an extension of the (Euclidean) halfspace depth hD from (1) will play a key role in the computation of ahD. First, define the empirical measure P in $\mathbb{R}^d$ as the sum of n Dirac measures at the data points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$ of $\boldsymbol{X}$ in $\mathbb{R}^d$. Denoting by $\mathcal{H}$ the set of all closed halfspaces in $\mathbb{R}^d$, the halfspace depth of $\boldsymbol{z}\in\mathbb{R}^d$ w.r.t. $\boldsymbol{X}$ can be rewritten as

$$hD\left(\boldsymbol{z}|\boldsymbol{X}\right)=\inf_{H\in\mathcal{H}:\,\boldsymbol{z}\in H}\#\left\{i:\boldsymbol{x}_i\in H\right\}=\inf_{H\in\mathcal{H}:\,\boldsymbol{z}\in H}P(H).$$

Plugging a signed (empirical) measure $P_\pm$ instead of P into the above expression, we define the *signed halfspace depth* of a point $\boldsymbol{z}\in\mathbb{R}^d$ w.r.t. a signed empirical measure $P_\pm$ in $\mathbb{R}^d$ to be

$$shD\left(\boldsymbol{z}|P_\pm\right)=\inf_{H\in\mathcal{H}:\,\boldsymbol{z}\in H}P_\pm(H). \tag{6}$$

The main difference between the signed halfspace depth shD and the ordinary halfspace depth hD is that in the computation of shD, one needs to take all halfspaces $H\in\mathcal{H}$ that contain $\boldsymbol{z}$, instead of only those H that contain $\boldsymbol{z}$ on its boundary. This comes naturally because we allow the points in $\mathbb{R}^d$ to attain also negative $P_\pm$-mass. Notice also that because the signed halfspace depth is evaluated w.r.t. a signed measure, we denote it $shD(\boldsymbol{z}|P_\pm)$ with the argument of the complete signed measure instead of just a dataset. Alternatively, we can also write $shD(\boldsymbol{z}|\left(\boldsymbol{y}_1,s_1\right),\ldots,\left(\boldsymbol{y}_m,s_m\right))$.

We are now ready to state the main result of the present section. It relates $ahD(\boldsymbol{z}|\boldsymbol{X})$ with shD of $\xi(\boldsymbol{z})$.

Theorem 3 *Let $\boldsymbol{X}$ be a dataset in $\mathbb{S}^{d-1}$ such that no point in $\boldsymbol{X}$ lies on the equator $\mathbb{S}_0^{d-1}$. Then for any $\boldsymbol{z}\in\mathbb{S}_+^{d-1}$ we can write*

$$ahD(\boldsymbol{z}|\boldsymbol{X})=\#\left\{i:\boldsymbol{x}_i\in\mathbb{S}_-^{d-1}\right\}+shD(\xi(\boldsymbol{z})|P_\pm), \tag{7}$$

with $P_\pm$ given by (4).

The complete proof of Theorem 3 is given in Appendix A. It is based on a simple observation: a point $\boldsymbol{x}\in\mathbb{S}_+^{d-1}$ lies in a halfspace $H_{\boldsymbol{0},\boldsymbol{p}}$ if and only if its gnomonic projection $\xi(\boldsymbol{x})$ lies in the $(d-1)$-dimensional halfspace $\mathcal{G}\cap H_{\boldsymbol{0},\boldsymbol{p}}$ (when considered inside $\mathcal{G}$). On the other hand, for points $\boldsymbol{x}\in\mathbb{S}_-^{d-1}$ in the opposite hemisphere, we have similarly that $\boldsymbol{x}$ is in the interior of $H_{\boldsymbol{0},\boldsymbol{p}}$ if and only if $\xi(\boldsymbol{x})$ lies in the complementary (relatively open) halfspace to $\mathcal{G}\cap H_{\boldsymbol{0},\boldsymbol{p}}$ in $\mathcal{G}$. Having expressed the number of data points inside every halfspace $H_{\boldsymbol{0},\boldsymbol{p}}$ in terms of the projected quantities $\xi(\boldsymbol{x})$ (and consequently also $P_\pm$), it remains to take an infimum over all halfspaces $H_{\boldsymbol{0},\boldsymbol{p}}$ containing $\boldsymbol{z}\in\mathbb{S}_+^{d-1}$; after some simplification, we arrive at (7).

In the rest of this article we canonically identify $\mathcal{G}$ with $\mathbb{R}^{d-1}$ and consider ξ as a function from $\mathbb{S}^{d-1}\setminus\mathbb{S}_0^{d-1}$ to $\mathbb{R}^{d-1}$.

2.2 Exact algorithm for $d = 1$

For $d = 1$ the unit sphere $\mathbb{S}^{d-1}$ is simply the set $\mathbb{S}^0 = \{-1, +1\}$. The angular halfspace depth of $z \in \mathbb{S}^0$ w.r.t a dataset X composed of points $x_1, \ldots, x_n \in \mathbb{S}^0$ is thus

$$ahD(z|x_1, \ldots, x_n) = \#\{i : z = x_i\}.$$

Complexity analysis

This trivial exact algorithm has complexity $\mathcal{O}(n)$.

2.3 Exact algorithm for $d = 2$

For data living on the unit circle $\mathbb{S}^1$, we use Theorem 3. We do so in four steps.

Algorithm Exact computation of $ahD(\boldsymbol{z}|\boldsymbol{X})$ for $\boldsymbol{z} \in \mathbb{S}^1$.

Step 1 **Projection** We project the data points from $\mathbb{S}^1$ to $\mathbb{R}$. The gnomonic projection is $\xi : \boldsymbol{y} \mapsto \xi(\boldsymbol{y}) : (y_1, y_2)' \mapsto y_1/y_2$. For each data point $\boldsymbol{x}_i$, we store both $x_i = \xi(\boldsymbol{x}_i) \in \mathbb{R}$ and its sign $s_i \in \{-1, 1\}$, which is the sign of the second coordinate of $\boldsymbol{x}_i$. The gnomonic transformation is also applied to $\boldsymbol{z}$; we obtain $z = \xi(\boldsymbol{z}) \in \mathbb{R}$. Denote by s_- the number of data points $\boldsymbol{x}_i$ such that $s_i = -1$.

Step 2 **Sorting and cleaning** We want to compute the signed halfspace depth of $z \in \mathbb{R}$ w.r.t. the (signed) dataset $(x_i, s_i) \in \mathbb{R} \times \{-1, 1\}$, $i = 1, \ldots, n$. The points x_i, $i = 1, \ldots, n$ are first sorted in ascending order. Then, they are adjusted for ties, that is, if $x_i = x_{i+1} = \cdots = x_{i+k}$, then the points $x_{i+1}, \ldots, x_{i+k}$ are deleted from the dataset, s_i is replaced by $\sum_{j=i}^{i+k} s_j$, and the points are relabeled. The data now consist of pairs (x_i, s_i), $i = 1, \ldots, m$, where s_i is an integer and $x_1 < x_2 < \cdots < x_m$.

Step 3 **Computing the signed halfspace depth** Define

$$U_i = \sum_{k=1}^{i} s_k, \quad i = 1, \ldots, m, \qquad \text{and}$$
$$V_i = \sum_{k=i}^{m} s_k, \quad i = 1, \ldots, m, \tag{8}$$

with $U_0 = 0$ and $V_{m+1} = 0$, and

$$\ell_z = \max\{i : x_i \le z\} \quad \text{and} \quad u_z = \min\{i : z \le x_i\}$$

where the conventions $\max \emptyset = 0$ and $\min \emptyset = m+1$ are used. Note that $\ell_z = u_z = i$ if $z = x_i$ for some i. Clearly,

$$P_\pm((-\infty, z]) = U_{\ell_z} \quad \text{and} \quad P_\pm([z, \infty)) = V_{u_z}.$$

If we take $y \ge z$, $P_\pm((-\infty, y])$ takes one of the values U_i, $i = \ell_z, \ldots, m$. Conversely, for $y \le z$, $P_\pm([y, \infty))$ takes one of the values V_i, $i = 1, \ldots, u_z$. Combining this gives

$$\begin{aligned} &shD(z|P_\pm) \\ &= \min\{\inf\{P_\pm((-\infty, y]) : y \ge z\}, \\ &\quad \inf\{P_\pm([y, \infty)) : y \le z\}\} \\ &= \min\{\min\{U_i : i = \ell_z, \ldots, m\}, \\ &\quad \min\{V_i : i = 1, \ldots, u_z\}\} \\ &= \min\{\min\{U_i : i = \ell_z, \ldots, m\}, \\ &\quad U_m - \max\{U_i : i = 0, \ldots, u_z - 1\}\}, \end{aligned} \tag{9}$$

where the last equality follows from $U_{i-1} + V_i = U_m$ and the convention $U_0 = 0$. We now abbreviate

$$U_i^{\uparrow min} = \min\{U_i, \ldots, U_m\} \quad \text{and}$$
$$U_i^{\downarrow max} = \max\{U_0, \ldots, U_i\}.$$

Therefore,

$$shD(z|P_\pm) = \min\left\{U_{\ell_z}^{\uparrow min}, U_m - U_{u_z-1}^{\downarrow max}\right\}.$$

Step 4 **Computing the angular halfspace depth** By Theorem 3 we have

$$ahD(\boldsymbol{z}|\boldsymbol{X}) = s_- + shD(z|P_\pm). \tag{10}$$

Implementation remarks

- Since there are only two points on the equator, namely $(1, 0)'$ and $(-1, 0)'$, **Step 1** can be modified to directly cope with data points lying on the equator. Simply map both points to $+\infty$ with sign $s = +1$ for $(1, 0)'$ and $s = -1$ for $(-1, 0)'$. This follows from the orthogonal invariance of the depth (Lemma 1) if we rotate all data points by a small positive angle ϵ.
- For points near the equator, the gnomonic projection in **Step 1** strongly inflates rounding errors that may be present in the data. Therefore, it is advisable to map $\mathbb{R} \cup \{+\infty\}$ to the bounded interval $(-\pi/2, +\pi/2]$ by applying the arctangent after the gnomonic projection. This is possible since the one-dimensional signed halfspace depth is invariant w.r.t. strictly increasing transformations.
- In **Step 4**, equation (10) gives the correct depth if the original point $\boldsymbol{z}$ is on the upper hemisphere or the "right-hand" point of the equator, $\mathbb{S}^1_+ \cup \{(1, 0)'\}$. If $\boldsymbol{z}$ lies in the lower hemisphere or the "left-hand" point of the equator, we reflect all points around the origin, i.e., we change the

sign of all data points. Using the notations

$$U_i^{\uparrow max} = \max\{U_i, \ldots, U_m\} \quad \text{and}$$
$$U_i^{\downarrow min} = \min\{U_0, \ldots, U_i\}$$

it is easy to derive that for points $z \in \mathbb{S}^1_- \cup \{(-1, 0)\}$ holds

$$ahD(z|\boldsymbol{X}) = s_+ - \max\left\{U_{\ell_z}^{\uparrow max}, U_m - U_{u_z-1}^{\downarrow min}\right\}$$

where s_+ is the number of data points $\boldsymbol{x}_i$ such that $s_i = +1$.

- In implementing **Step 3**, there are two possible strategies:
  - **Algorithm `aHD2(s)`.** The first strategy should be advantageous when only the depth of a single point z (or a small number of depths) has to be computed. In that case, in **Step 3**, we only compute the values $U_i^{\uparrow min}$, $U_i^{\uparrow max}$, $U_i^{\downarrow min}$, $U_i^{\downarrow max}$ that are actually needed.
  - **Algorithm `aHD2(m)`.** The second strategy is superior when a large number of depths have to be computed. In that case, in **Step 3** we compute all values $U_i^{\uparrow min}$, $U_i^{\uparrow max}$, $U_i^{\downarrow min}$, $U_i^{\downarrow max}$, $i = 0, \ldots, m$, directly and store them in four arrays.

Complexity analysis

Projecting the n data points in **Step 1** has a complexity of $\mathcal{O}(n)$. In **Step 2** we have to sort the n data points, which has complexity $\mathcal{O}(n \log n)$, while cleaning can be done in $\mathcal{O}(n)$. Computing the quantities U_i, the indices ℓ_z, u_z, and the minima and maxima $U_i^{\uparrow min}$, $U_i^{\uparrow max}$, $U_i^{\downarrow min}$, $U_i^{\downarrow max}$ as well as the signed halfspace depth in **Step 3** is $\mathcal{O}(n)$. Finally, **Step 4** is $\mathcal{O}(1)$. This results in an overall complexity of $\mathcal{O}(n \log n)$, regardless of whether variant `aHD2(s)` or `aHD2(m)` is used.

2.4 Exact algorithm for $d = 3$

We use Theorem 3 again; the pseudocode of the algorithm is described in **Steps 1–4** below.

Algorithm Exact computation of $ahD(z|\boldsymbol{X})$ for $z \in \mathbb{S}^2$.

Step 1. Projection The data are projected to $\mathbb{R}^2$ using the gnomonic projection $\xi\colon \boldsymbol{y} \mapsto \xi(\boldsymbol{y})\colon (y_1, y_2, y_3)' \mapsto (y_1/y_3, y_2/y_3)'$. For each data point $\boldsymbol{x}_i$ we store both $\xi(\boldsymbol{x}_i) \in \mathbb{R}^2$ and its sign $s_i \in \{-1, 1\}$ defined as the sign of the third coordinate of $\boldsymbol{x}_i$. The gnomonic projection is also applied to z to obtain $\xi(z) \in \mathbb{R}^2$. Store the number of data points $\boldsymbol{x}_i$ such that $s_i = -1$ in a variable `sminus`.
After the gnomonic projection, we no longer need to work with the original data points $\boldsymbol{x}_i \in \mathbb{S}^2$ or $z \in \mathbb{S}^2$. In what follows we therefore abuse notation and simply write $\boldsymbol{x}_i \in \mathbb{R}^2$ instead of $\xi(\boldsymbol{x}_i) \in \mathbb{R}^2$, $i = 1, \ldots, n$, and $z \in \mathbb{R}^2$ instead of $\xi(z) \in \mathbb{R}^2$. In the new notation, our task is to compute the signed halfspace depth of z w.r.t. the signed dataset $(\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{x}_n, s_n)$ in $\mathbb{R}^2$.

Step 2. Augmentation, sorting and cleaning If the (projected) data point z is different from all (projected) data points $\boldsymbol{x}_i, i = 1, \ldots, n$, we augment the (signed) dataset by appending $(\boldsymbol{x}_0, s_0) = (z, 0)$.
For reasons that will become clear in **Step 3.1** below, the augmented dataset is sorted according to the order $\prec$ defined by

$$\boldsymbol{a} = (a_1, a_2)' \prec \boldsymbol{b} = (b_1, b_2)' \iff (a_1 < b_1) \text{ or } (a_1 = b_1 \text{ and } a_2 > b_2).$$

The augmented and sorted dataset is cleaned as in **Step 2** in Section 2.3; only the $m \le n + 1$ unique values of $\boldsymbol{x}_i$ are considered, redundant points are deleted from the dataset, and the associated signs are adjusted. Now, we may assume that $\boldsymbol{x}_i$, $i = 1, \ldots, m$ are pairwise distinct and sorted in increasing order, $\boldsymbol{x}_1 \prec \ldots \prec \boldsymbol{x}_m$, z is equal to some $\boldsymbol{x}_i$, and all $\boldsymbol{x}_i$, $i = 1, \ldots, m$, are associated with appropriate integer signs s_i.

Step 3. Computing the signed halfspace depth The signed halfspace depth (6) satisfies the projection property (Dyckerhoff 2004), meaning that for the depth in $\mathbb{R}^{d-1}$ we can write

$$shD\left(z|(\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{x}_m, s_m)\right) = \inf_{\boldsymbol{p} \in \mathbb{S}^{d-2}} shD(\boldsymbol{p}'z|(\boldsymbol{p}'\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{p}'\boldsymbol{x}_m, s_m)). \quad (11)$$

Using an argument quite analogous to Lemma 7, it is easy to show that for $d = 3$, in the infimum on the right-hand side of (11), we can restrict the directions $\boldsymbol{p} \in \mathbb{S}^{d-2} = \mathbb{S}^1$ to those such that the projected points $\boldsymbol{p}'\boldsymbol{x}_i, i = 1, \ldots, m$, are pairwise distinct. For such a direction $\boldsymbol{p}$ let $\eta_{\boldsymbol{p}}$ be the permutation of the indices $\{1, \ldots, m\}$ for which

$$\boldsymbol{p}'\boldsymbol{x}_{\eta_{\boldsymbol{p}}(1)} < \cdots < \boldsymbol{p}'\boldsymbol{x}_{\eta_{\boldsymbol{p}}(m)}. \quad (12)$$

As is clear from the discussion in **Step 3** in Section 2.3, the univariate signed halfspace depth $shD\left(\boldsymbol{p}'z|(\boldsymbol{p}'\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{p}'\boldsymbol{x}_m, s_m)\right)$ can be

determined from the signs $s_1, \ldots, s_m$ and the permutation $\eta_{\boldsymbol{p}}$ only. Indeed, for a permutation η of $\{1, \ldots, m\}$ we define

$$U_0^\eta = 0 \quad \text{and} \quad U_i^\eta = \sum_{k=1}^{i} s_{\eta(k)}, \quad i = 1, \ldots, m,$$

and let $r(\eta) = r(\boldsymbol{z}, \eta)$ be the rank of $\boldsymbol{p}'\boldsymbol{z}$ in the ordered sequence $\boldsymbol{p}'\boldsymbol{x}_{\eta_{\boldsymbol{p}}(1)}, \ldots, \boldsymbol{p}'\boldsymbol{x}_{\eta_{\boldsymbol{p}}(m)}$. Then, from (9) and (11) it follows that

$$shD\,(\boldsymbol{z}|(\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{x}_m, s_m)) = \inf_{\boldsymbol{p}\in\mathbb{S}^1:\ \eta_{\boldsymbol{p}} \text{ encodes (12)}} \min\left\{ \min_{i\geq r(\eta_{\boldsymbol{p}})} U_i^{\eta_{\boldsymbol{p}}}, U_m^{\eta_{\boldsymbol{p}}} - \max_{i<r(\eta_{\boldsymbol{p}})} U_i^{\eta_{\boldsymbol{p}}} \right\}. \tag{13}$$

As observed, e.g., by Edelsbrunner (1987, Observation 2.1), there exists a finite number ($m(m-1)$ if the data are in general position, less than $m(m-1)$ otherwise) of permutations $\eta_{\boldsymbol{p}}$ encoding all possible orderings in (12) with $\boldsymbol{p} \in \mathbb{S}^1$. An efficient enumeration of all such permutations can be achieved using a so-called *circular sequence* of permutations, see Edelsbrunner (1987, Chap. 2). Circular sequences have proved to be fruitful in designing algorithms for computing certain depths in dimension $d = 2$, see Ruts and Rousseeuw (1996), Dyckerhoff (2000), Cascos (2007), or Cascos and Ochoa (2021). The key idea is that the order of two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is reversed in the ordering (12) with changing $\boldsymbol{p}$ if and only if a straight line through the origin in the direction $\boldsymbol{p}$ is orthogonal to the line joining $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, see Figure 4.

Denote by $\mathcal{C}$ the set of all permutations in the circular sequence corresponding to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. By interchanging the minima in (13) we get

$$\begin{aligned} &shD\,(\boldsymbol{z}|(\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{x}_m, s_m)) \\ &= \min_{\eta\in\mathcal{C}} \min\left\{ \min_{i\geq r(\eta)} U_i^{\eta}, U_m^{\eta} - \max_{i<r(\eta)} U_i^{\eta} \right\} \\ &= \min\left\{ \min_{\eta\in\mathcal{C}, i\geq r(\eta)} U_i^{\eta}, U_m^{\eta} - \max_{\eta\in\mathcal{C}, i<r(\eta)} U_i^{\eta} \right\}. \end{aligned} \tag{14}$$

In the following steps, we explain how to use the idea of circular sequence to evaluate the right-hand side of (14).

Step 3.1. **Initialization** For the initial permutation in the circular sequence, we choose direction $\boldsymbol{p}_0 = (1, 0)'$. This orders the points $\boldsymbol{x}_i$ according to their first coordinate. To break potential ties, we tilt $\boldsymbol{p}_0$ slightly to $\boldsymbol{p}_{-\epsilon} = (\cos(-\epsilon), \sin(-\epsilon))'$ for $\epsilon > 0$ sufficiently small. This gives exactly the order $\prec$ in which the points were already sorted in **Step 2**, Hence, $\eta_{\boldsymbol{p}_{-\epsilon}}(i) = i$ for all i.

To keep track of the current permutation, we store the inverse permutation η^{-1}, i.e., the ranks of the points $\boldsymbol{x}_i$, in an array `rank` of length m which is initialized as `rank[i]:=i`. We also initialize the cumulative sums $U_i^\eta, i = 1, \ldots, m$ and store them in an array `U` of length m. The variables `minU` and `maxU` are initialized to ∞ and $-\infty$, respectively.

Step 3.2. **Setting up the circular sequence** For each pair of indices $k < \ell$, compute the angle $\gamma_{k,\ell} \in [0, \pi)$ of the line passing through the origin that is orthogonal to $\boldsymbol{x}_k - \boldsymbol{x}_\ell$. Store all these angles together with the indices k and ℓ in an array `angles` of length $\binom{m}{2}$. Sort the angles in ascending order. Some of the angles may be tied. Break those ties according to the index of the first point. If these indices are again tied, use the index of the second point to break ties.

Step 3.3. **Passing through the circular sequence** We go through a loop over all angles $\gamma_{k,\ell}$ sorted in increasing order. In **Steps 3.3.a** and **3.3.b** we explain how (i) the ranks of the points in the array `rank`, (ii) the cumulative sums U_i^η, $i = 1, \ldots, m$ in array `U`, and (iii) the variables `minU` and `maxU` are updated in each move from one permutation to the next one in the circular sequence.

Step 3.3.a. **Strong general position** For starters, assume that the data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are in a *strong general position*, meaning that

(1) on each straight line in $\mathbb{R}^2$ there are at most two data points,
(2) there is no pair of parallel straight lines such that there are two data points on each of these lines.

As we pass through the angle $\gamma_{k,\ell}$, the permutation changes from, say, η_- to η_+. Note that $\boldsymbol{x}_k$ and $\boldsymbol{x}_\ell$ swap order from η_- to η_+. Thus, their ranks will be adjacent in both these permutations. Since initially $\boldsymbol{x}_k \prec \boldsymbol{x}_\ell$, it holds that

$$\eta_-^{-1}(k) = r,\ \eta_-^{-1}(\ell) = r+1 \quad \text{and} \quad \eta_+^{-1}(k) = r+1,\ \eta_+^{-1}(\ell) = r$$

for some r. Thus, for updating the permutation, we exchange the ranks of $\boldsymbol{x}_k$ and $\boldsymbol{x}_\ell$ by performing `swap(rank[k], rank[l])`. Since the permutations η^- and η^+ differ only in the arguments r and $r+1$, the cumulative sums $U_i^{\eta^-}$ and

$U_i^{\eta+}$ will differ only for $i = r$. More specifically,

$$U_r^{\eta+} = U_r^{\eta-} + \left(s_{\eta+(r)} - s_{\eta-(r)}\right) = U_r^{\eta-} + (s_\ell - s_k)\,.$$

This shows that only one of the cumulative sums U_i^η, $i = 1, \ldots, m$, needs to be updated in a single move. Therefore, we update the element `U[r]` of the array `U`. Depending on the position of r relative to $r(\eta_+)$ (the rank of $\boldsymbol{p}'\boldsymbol{z}$ in the ordered sequence $\boldsymbol{p}'\boldsymbol{x}_{\eta_+(1)}, \ldots, \boldsymbol{p}'\boldsymbol{x}_{\eta_+(m)}$), we then update exactly one of the variables `minU` and `maxU`: If $r \geq r(\eta_+)$, then `minU:=min (minU, U[r])`, if $r < r(\eta_+)$, then `maxU:= max(maxU, U[r])`.

Step 3.3.b. **General situation** Now assume that the points are not in a strong general position. Then, several sets of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ may lie on parallel straight lines. In Figure 4 we have three sets of collinear points, namely $\{\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_5\}$, $\{\boldsymbol{x}_2, \boldsymbol{x}_7\}$, and $\{\boldsymbol{x}_4, \boldsymbol{x}_6, \boldsymbol{x}_8\}$. All three sets lie on lines that are orthogonal to a straight line through the origin, having an angle of $\pi/4$ with the positive horizontal axis. In the left-hand panel of Figure 4, the projection line has an angle of less than $\pi/4$, giving the order 1, 3, 5, 2, 7, 4, 6, 8. In the right-hand panel, the projection line has an angle greater than $\pi/4$, giving the order 5, 3, 1, 7, 2, 8, 6, 4. Thus, in every subset, the order has been reversed.

If we use the tie-breaking scheme for the angles introduced in **Step 3.2**, we can simply go through all the angles that are tied and update the array `rank`, as well as the cumulative sums `U`, as described in **Step 3.3.a**, to get the correct ordering of points. However, since all these changes take place simultaneously, the update of the variables `minU` and `maxU` has to be postponed until all the tied angles are processed. Therefore, we have to keep track of the affected positions. For that, we use a set-valued variable `positions` which is initialized to the empty set. For each tied angle, we update `positions` by executing `positions:=setunion(positions,r)` where `r` is the position that has changed in this step. Once all tied angles are processed, we update `minU` and `maxU` by going through all the affected positions as in **Step 3.2.a**. After that, the variable `positions` is again initialized to the empty set.

The above updates **Steps 3.3.a** or **3.3.b** are done for all $\binom{m}{2}$ angles $\gamma_{k,\ell}$ in the for-loop in **Step 3.3**.

Step 4. **End of the algorithm** The resulting signed halfspace depth of $\boldsymbol{z}$ is returned as `min(minU, sum(U) - maxU)`, where `sum(U)` is the sum of all the elements of `U`. The final angular halfspace depth of the original direction $\boldsymbol{z} \in \mathbb{S}^2$ is expressed as `sminus + min(minU, sum(U) - maxU)`, thanks to Theorem 3.

Implementation remarks

Since the case when the points are in a strong general position (**Step 3.3.a**) is much easier to implement than the general case (**Step 3.3.b**), we have implemented both versions as `aHD3(GP)` and `aHD3(nGP)`, respectively.

Complexity analysis

Projecting the data points into the plane in **Step 1** has time complexity $\mathcal{O}(n)$. Whereas augmentation and cleaning in **Step 2** are of complexity $\mathcal{O}(n)$, the sorting has a complexity $\mathcal{O}(n \log n)$. For computing the signed halfspace depth in $\mathbb{R}^2$, the initialization phase in **Step 3.1** is $\mathcal{O}(n)$. In **Step 3.2** the value m will be at most $n + 1$. Processing all pairs $(\boldsymbol{x}_k, \boldsymbol{x}_\ell)$ has a complexity of $\mathcal{O}(n^2)$ and the sorting of the angles is $\mathcal{O}(n^2 \log n)$. In **Step 3.3**, we go through all the elements of the circular sequence. Therefore, the complexity is $\mathcal{O}(n^2)$ times the complexity of a single update step. From the discussion in **Steps 3.3.a** and **3.3.b** it follows that a single update step has only complexity $\mathcal{O}(1)$. Therefore, the complexity of **Step 3.3** is $\mathcal{O}(n^2)$. **Step 4** has complexity $\mathcal{O}(1)$. Putting things together gives the exact algorithm an overall complexity of $\mathcal{O}(n^2 \log n)$.

2.5 Complexity analysis: Depth of multiple points

With our exact algorithms, we have so far discussed only the time complexity of computing ahD of a single point $\boldsymbol{z}$. In data analysis, it is typically of interest to compute the depths of a large number of points simultaneously. We will have a closer look at how the exact algorithms for $d = 1, 2, 3$ perform in this respect. Assume that we have n data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, and m points $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m$ whose depth we want to compute. We will also report the complexity for the special case in which the depths of all data points in $\boldsymbol{X}$ must be computed, in which case we have $m = n$.

Complexity analysis for $d = 1$

Trivially, the complexity is $\mathcal{O}(n + m)$. For $m = n$, we get the complexity of $\mathcal{O}(n)$.

Complexity analysis for $d = 2$

The projection **Step 1** has order $\mathcal{O}(n+m)$. In **Step 2**, we only sort the points $\boldsymbol{x}_i$, which gives a complexity of $\mathcal{O}(n \log n)$.

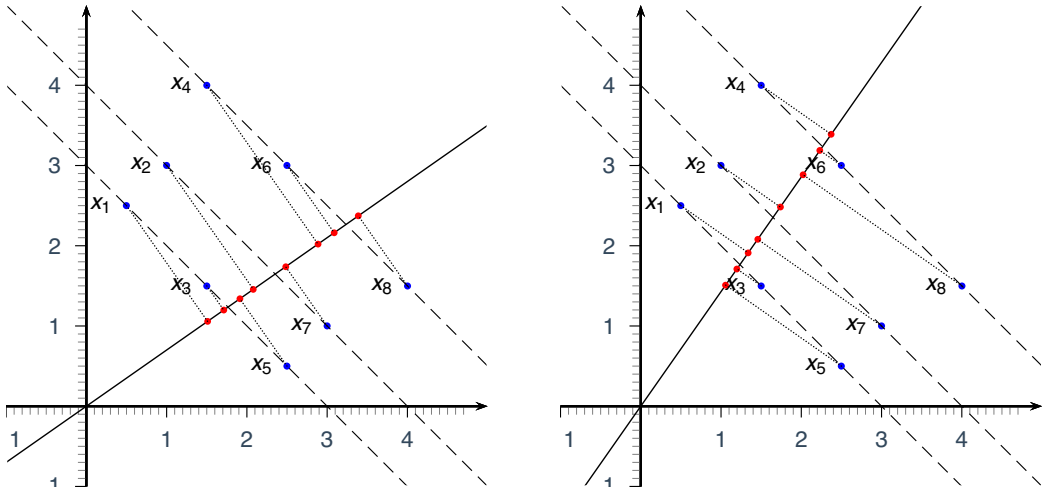


Fig. 4 A single move in a circular sequence when points are not in a strong general position. For a detailed description, see **Step 3.3.b** in Section 2.4

In addition, cleaning has also to be done only for the $\boldsymbol{x}_i$'s, which is again $\mathcal{O}(n)$. In **Step 3** we have to distinguish whether variant `aHD2(s)` or `aHD2(m)` is used. If `aHD2(s)` is used, the cumulative sums (8) have to be calculated only once, which has complexity $\mathcal{O}(n)$. Then, for each of the m points $\boldsymbol{z}_j$ we have to compute the indices ℓ_z, u_z ($\mathcal{O}(\log n)$ when the binary search is applied) and the minimum and maximum in (9) ($\mathcal{O}(n)$). This yields for **Step 3** a complexity of $\mathcal{O}(mn)$. If variant `aHD2(m)` is used, the cumulative sums (8) and all the cumulative minima and maxima are calculated once, which has complexity $\mathcal{O}(n)$. Then, for each of the m points $\boldsymbol{z}_j$ we only have to compute the indices ℓ_z, u_z which has complexity $\mathcal{O}(\log n)$ as stated above. Therefore, the complexity of **Step 3** is $\mathcal{O}(n+m\log n)$. Finally, **Step 4** is $\mathcal{O}(m)$. Hence, the overall complexity is $\mathcal{O}(n\log n + mn)$ for variant `aHD2(s)` and $\mathcal{O}(n\log n + m\log n)$ for variant `aHD2(m)`. For the special case in which the depths of all data points $\boldsymbol{x}_i$ have to be computed, that is, $m = n$, we get $\mathcal{O}(n^2)$ for `aHD2(s)` and $\mathcal{O}(n\log n)$ for `aHD2(m)`.

Complexity analysis for $d = 3$

The projection in **Step 1** has complexity $\mathcal{O}(n+m)$ again. In **Step 2** we do the sorting and cleaning only for the n data points $\boldsymbol{x}_i$ which gives a complexity of $\mathcal{O}(n\log n)$. Then, for the computation of the signed halfspace depth we start with sorting only the angles $\gamma_{k,\ell}$ which are defined by pairs $\boldsymbol{x}_k, \boldsymbol{x}_\ell$ in **Step 3.2** (complexity $\mathcal{O}(n^2\log n)$). Then again, we process all points $\boldsymbol{z}_j$ separately. Inserting $\boldsymbol{x}_0 = \boldsymbol{z}$ into the dataset $\boldsymbol{x}_i$, $i = 1,\ldots,n$, is done in linear time $\mathcal{O}(n)$. Now we also have to consider the angles defined by $\boldsymbol{x}_0$ and one of the points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$. Computing these angles in **Step 3.2** takes complexity $\mathcal{O}(n)$. Next we sort these n angles $\gamma_{0,\ell}$ separately (complexity $\mathcal{O}(n\log n)$), and merge the two sorted sequences ($\binom{n}{2}$ angles $\gamma_{k,\ell}$ with $k, \ell > 0$ and n values $\gamma_{0,\ell}$ with $\ell > 0$) into a single list (complexity $\mathcal{O}(\binom{n}{2} + n) = \mathcal{O}(n^2)$). In **Step 3.3**, we have to iterate through the array of all angles, which consists of no more than $\binom{n+1}{2}$ angles. Therefore, this step has complexity $\mathcal{O}(n^2)$ again. Finally, **Step 4** has complexity $\mathcal{O}(m)$. Putting all together, the overall complexity is given by $\mathcal{O}(n^2\log n + mn^2)$. For the special case $m = n$ we get $\mathcal{O}(n^3)$.

Summary

As our analysis shows, we should see a significant reduction in computation times when several depths are computed simultaneously w.r.t. the same dataset $\boldsymbol{X}$. This situation is quite exceptional for the angular halfspace depth; analogous reduction of computational complexity is not possible for the exact (Euclidean) halfspace depth using the algorithm of Dyckerhoff and Mozharovskyi (2016), see the discussion at the end of Section 3. We will further discuss this remarkable feature of the angular halfspace depth in the numerical study in Section 4.

2.6 Approximate computation: Random angular halfspace depth

Before proceeding with a series of rather elaborate exact computation algorithms for ahD in dimension $d > 3$, we note that our advances in Section 2.3 and 2.4 also offer simple approximate procedures for computing ahD. The simplest approximation scheme is the *random search method* (Dyckerhoff 2004, Section 6) that we describe now. We have seen in (11) that the signed halfspace depth satisfies the projection property. On the right-hand side of (11) is shD on the real line $\mathbb{R}$. The idea of the random search is, in $\mathbb{R}^{d-1}$, instead of searching for the minimum in (11) in the whole set of directions $\boldsymbol{p} \in \mathbb{S}^{d-2}$, to take a sufficiently large collec-

tion of N randomly chosen, uniformly distributed directions $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N \in \mathbb{S}^{d-2}$, and approximate (11) by the minimum

$$shD_N\,(\boldsymbol{z}|(\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{x}_n, s_n)) = \min_{j=1,\ldots,N} shD\left(\boldsymbol{p}_j'\boldsymbol{z}|(\boldsymbol{p}_j'\boldsymbol{x}_1, s_1), \ldots, (\boldsymbol{p}_j'\boldsymbol{x}_n, s_n)\right). \tag{15}$$

Here, the univariate shD is already simple to compute, by means of (9) in Section 2.3. An alternative approximation procedure would be using random projections into two-dimensional subspaces of $\mathbb{R}^{d-1}$ and the exact computation of the signed halfspace depth in $\mathbb{R}^2$ as proposed in Section 2.4. Having computed an approximate signed halfspace depth in (15), Theorem 3 is used to approximate ahD.

Complexity analysis

Both these approximation procedures are quite fast even in higher dimensions $d > 3$ provided N is kept fixed. For the methods based on (15) with projections into $\mathbb{R}$, the computational complexity is only $\mathcal{O}(n\,N)$; projecting N times into $\mathbb{R}^2$ results in complexity $\mathcal{O}(n\log(n)\,N)$. As long as N is fixed, these results are independent of the dimension d. Thus, thanks to Theorem 3, we devise fast approximate algorithms for ahD. In the case of (Euclidean) halfspace depth, such approximate procedures have been studied in Nagy et al. (2020) and Dyckerhoff et al. (2021), where it was shown that, despite being extremely fast, their performance is rather poor, especially in higher dimensions d. Their performance cannot be expected to be better for the angular halfspace depth. Thus, these simple approximations must be used with caution. To ensure a stable approximation accuracy in high-dimensional settings, the number of random projections must be increased accordingly as the dimension rises. Consequently, the overall computational complexity and the accuracy of these algorithms would be primarily determined by the rate at which the number of projections grows with the dimension.

Since a comprehensive comparison of approximate algorithms would exceed the scope of this article, we have chosen to focus exclusively on exact algorithms in the sequel. A thorough investigation of approximate algorithms will be a subject of further study.

3 Exact computation of *ahD* in dimension $d > 3$

Having designed efficient algorithms for ahD in dimensions $d = 1, 2, 3$ in Section 2, we now turn to the problem of computing ahD exactly in any dimension $d > 3$. We proceed recursively, by decreasing the dimension d of the problem, until we can use the specialized algorithms for $d = 1, 2, 3$ proposed above. This approach is similar to that used for the (Euclidean) hD in Dyckerhoff and Mozharovskyi (2016); the end of this section is dedicated to a detailed discussion of the main similarities and differences of the two approaches.

Our intention is to develop a family of computation schemes that allow us to reduce the dimensionality d of computing $ahD(\boldsymbol{z}|\boldsymbol{X})$. It will be convenient to assume that $\boldsymbol{z} \notin \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. If some of the data points of $\boldsymbol{X}$ are equal to $\boldsymbol{z}$, these points are removed from the dataset, and their number is added to the angular halfspace depth (2) of $\boldsymbol{z}$ w.r.t. the remaining points. We will further assume that $\text{span}\,(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \mathbb{R}^d$, that is, the data is full-dimensional. If this is not the case, i.e., if the data points are contained in some k-dimensional subspace of $\mathbb{R}^d$, $k < d$, we canonically identify $S = \text{span}\,(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ with $\mathbb{R}^k$. If $\boldsymbol{z}$ is also contained in this subspace, we directly proceed with the computation of ahD in $\mathbb{S}^{k-1}$. If $\boldsymbol{z}$ does not lie in this k-dimensional subspace S, it can be shown that $ahD(\boldsymbol{z}|\boldsymbol{X}) = hD(\boldsymbol{0}|\tilde{\boldsymbol{X}})$ where $\tilde{\boldsymbol{X}}$ denotes the dataset $\boldsymbol{X}$ projected to S. This can be seen, e.g., from Nagy and Laketa (2025, Theorem 6).

To introduce our procedure, we need additional notation. For a set $I = \{i_1, \ldots, i_k\}$ of indices of data points we denote by

$$I^* = \{j : \boldsymbol{x}_j \in \text{span}\left(\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}\right)\}$$

the set of all indices $j = 1, \ldots, n$ such that $\boldsymbol{x}_j$ is contained in the linear hull of $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$. Obviously, $I \subseteq I^*$; in the common situation of $k \leq d$ data points lying in a general position with the origin,[4] we naturally have $I = I^*$.

Our main result is motivated by the exact computation of the (Euclidean) halfspace depth hD from Dyckerhoff and Mozharovskyi (2016, Theorem 2). In what follows, we adapt that procedure to directional data. We reduce the calculation of ahD in dimension d to a repeated calculation of ahD in lower dimensions. For the lower-dimensional depth we, however, have to work in a slightly more general setup of data points that do not necessarily lie on the unit sphere.

[4] Recall that a set S of points in $\mathbb{R}^d$ is said to lie in *general position* if no subset of k of these points lies in a $(k-2)$-dimensional affine space, for all $k = 2, \ldots, d+1$. If there are $n > d$ points in S, this is equivalent to saying that no hyperplane in $\mathbb{R}^d$ contains more than d points from S. We say that a set S lies in *general position with the origin* $\boldsymbol{0} \in \mathbb{R}^d$ if $S \cup \boldsymbol{0}$ lies in general position.

Remark 4 In the definition of the angular halfspace depth (2), it is not necessary to restrict z and $x_1, \ldots, x_n$ to the unit sphere $\mathbb{S}^{d-1}$. Obviously, for $\lambda_0, \lambda_1, \ldots, \lambda_n > 0$ it holds that

$$\begin{aligned} &ahD(\lambda_0 z|\lambda_1 x_1, \ldots, \lambda_n x_n) \\ &= \min_{p\in\mathbb{S}^{d-1}:\,(\lambda_0 z)\in H_{0,p}} \#\{i:\, p'(\lambda_i x_i) \geq 0\} \\ &= \min_{p\in\mathbb{S}^{d-1}:\, z\in H_{0,p}} \#\{i:\, p' x_i \geq 0\} = ahD(z|x_1, \ldots, x_n). \end{aligned} \tag{16}$$

Therefore, the angular halfspace depth depends only on the directions (not the magnitude) of the vectors z and $x_1, \ldots, x_n$ in $\mathbb{R}^d$. All of these vectors can be replaced by strictly positive multiples without altering the depth. This allows us to write $ahD(z|X)$ also for z and data points X in $\mathbb{R}^d \setminus \{0\}$, which is well defined by (16).

The following theorem is the main result of the present section.

Theorem 5 *For $k = 1, \ldots, d-1$ denote by $\mathcal{L}_k$ the set of all subsets $I = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ of cardinality k such that (i) the linear space $S_I = \operatorname{span}(x_{i_1}, \ldots, x_{i_k})$ has dimension k (that is, the points $(x_i)_{i\in I}$ are linearly independent) and (ii) $z \notin S_I$. Denote by $p_1, \ldots, p_k$ an orthonormal basis of the space S_I, and write $P_I = [p_1, \ldots, p_k] \in \mathbb{R}^{d\times k}$ for the matrix with columns p_i, $i = 1, \ldots, k$. Similarly, let $q_1, \ldots, q_{d-k}$ be an orthonormal basis of the orthogonal complement $S_I^\perp$ of S_I, and let $Q_I = [q_1, \ldots, q_{d-k}] \in \mathbb{R}^{d\times(d-k)}$. Then we can write*

$$\begin{aligned} &ahD(z|X) \\ &= \min_{I\in\mathcal{L}_k}\Big[ahD(Q_I' z|Q_I' X_{(I^*)^c}) + hD(0|P_I' X_{I^*})\Big]. \end{aligned} \tag{17}$$

The proof of our main result is technical. It requires several auxiliary lemmas and is relegated to Appendix B. Theorem 5 reduces the computation of ahD in dimension d to a repeated computation of (i) the angular halfspace depth in dimension $d-k$, and (ii) the (Euclidean) halfspace depth in dimension k. Each of these depths is evaluated w.r.t. a different dataset; $(I^*)^c$ in (17), of course, means the complement of the set I^* in $\{1, \ldots, n\}$. An important special case of the algorithm arises when the data points $x_1, \ldots, x_n$ and 0 are in general position. In that case, $I^* = I$ for each index set $I \in \mathcal{L}_k$, meaning that X_{I^*} contains exactly k data points. In addition, $hD(0|P_I' X_{I^*}) = 0$, which can be seen by choosing the halfspace $H_{0,p}$ with p such that $p'(P_I' x_{i_1} - P_I' x_{i_j}) = 0$ for $j = 2, \ldots, k$. Therefore, we obtain the following corollary.

Corollary 6 *If the points $x_1, \ldots, x_n$ and 0 are in general position, then for each $k = 1, \ldots, d-1$ we can express*

$$ahD(z|X) = \min_{I\in\mathcal{L}_k} ahD(Q_I' z|Q_I' X_{I^c}).$$

For implementing formula (17), we need to compute the orthogonal projection matrices P_I and Q_I. This is done using the *QR factorization*, see, e.g., Golub and Van Loan (2013, Chap. 5). The exact computation of the (Euclidean) halfspace depth $hD(0|P_I' X_{I^*})$ in (17) is performed using the procedures from Dyckerhoff and Mozharovskyi (2016) implemented in the R package **ddalpha** (Pokotylo et al. 2019). Finally, for computing the angular halfspace depth $ahD(Q_I' z|Q_I' X_{(I^*)^c})$ in (17), we need an efficient exact algorithm for ahD in low dimensions d so that the reduction process can be stopped. We use our algorithms for $d = 1, 2, 3$ from Section 2.

Combinatorial and recursive algorithms

In Theorem 5, we choose the tuning parameter $k = 1, \ldots, d-1$. Many different strategies are conceivable. The one extreme is when the dimensionality is reduced in a single step to $d = 1, 2, 3$. Immediately after this reduction, one of the specialized algorithms for low dimensions from Section 2 is used. We call these algorithms *combinatorial algorithms* and denote them by `C1`, `C2` and `C3`, respectively, according to the dimension d of the space we project to. If necessary, we distinguish between the variants of the exact algorithm used and write `C2(S)` and `C2(M)` (for dimension $d = 2$) and `C3(NGP)` and `C3(GP)` (for dimension $d = 3$). The other extreme is when the dimensionality is reduced only by one in each step. We call these algorithms *recursive algorithms*. Again we have to decide when to stop the recursion and apply a specialized algorithm from Section 2. We denote these algorithms by `R1`, `R2(S)`, `R2(M)`, `R3(NGP)`, and `R3(GP)`.

Complexity analysis

For analyzing the complexity of our general algorithms for $d > 3$ we will, for simplicity, assume that the data are in a general position. In that case, we follow Corollary 6. We directly consider the same situation as in Section 2.5, that is, the depths of m points $z_1, \ldots, z_m$ w.r.t. a common dataset $x_1, \ldots, x_n$ have to be computed.

Complexity of combinatorial algorithms

In the combinatorial algorithms, the dimensionality is reduced in a single step to $k = 1, 2$, or 3. Therefore, we have to enumerate all $(d-k)$-element subsets of data points $x_1, \ldots, x_n$ in $\mathcal{L}_{d-k}$, which is of order $\mathcal{O}(n^{d-k})$. Then, for each subset of $d-k$ data points indexed by $I \in \mathcal{L}_{d-k}$, we use QR factorization to obtain Q_I. This QR factorization is independent of n and is therefore $\mathcal{O}(1)$. The projection of the data points and the points $z_1, \ldots, z_m$ to a space of dimension k has complexity $\mathcal{O}(n+m)$. Finally, the computation

of the angular halfspace depth is of order $\mathcal{O}(n+m)$ (for $k=1$), $\mathcal{O}(n\log n+mn)$ (for $k=2$, variant `aHD2(s)`), $\mathcal{O}(n\log n+m\log n)$ (for $k=2$, variant `aHD2(m)`), and finally $\mathcal{O}(n^2\log n+mn^2)$ (for $k=3$, both variants), as we saw in Section 2. Therefore, for each considered subset, the time complexity is given by the time complexity of the exact algorithm used. Consequently, the overall complexity of the combinatorial algorithms is $\mathcal{O}(n^d+mn^{d-1})$ for `C1`, $\mathcal{O}(n^{d-1}\log n+mn^{d-1})$ for `C2(S)` and both variants of `C3`, $\mathcal{O}(n^{d-1}\log n+mn^{d-2}\log n)$ for `C2(M)`.

Complexity of recursive algorithms

In the recursive algorithms, we reduce the dimensionality only by one in each step until we reach dimension $k=1,2$, or 3. That means that in the main loop, we iterate through $\mathcal{L}_1$, which contains all n data points. That gives a complexity of $\mathcal{O}(n)$. For each index $I=\{i\}\in\mathcal{L}_1$ of a data point $\boldsymbol{x}_i$, we project the data points $\boldsymbol{x}_i$ as well as the points $\boldsymbol{z}_j$ on the hyperplane $S_I^\perp$ orthogonal to $\boldsymbol{x}_i$ and compute the angular halfspace depth in dimension one less. Therefore, for each performed recursion step, the complexity is multiplied by n. We obtain the same complexities as for the combinatorial algorithms.

Summary

Our analysis shows that the algorithms `C1` and `R1` are inferior to `C2(S)`, `R2(S)`, `C3` and `R3`, and these are inferior to `C2(M)` and `R2(M)`. We will confirm this finding also in the numerical study in Section 4. Table 1 summarizes the time complexities of the combinatorial algorithms in the three considered scenarios: computing the depth of a single point, computing the depth of multiple points, and computing the depth of all points in the dataset itself, as discussed in Section 2.5. Observe that the rates of computing $ahD(\boldsymbol{z}|\boldsymbol{X})$ for a single point $\boldsymbol{z}\in\mathbb{S}^{d-1}$ are the same as the corresponding rates for computing hD in $\mathbb{R}^d$ from Dyckerhoff and Mozharovskyi (2016). For the computation of m depths simultaneously, ahD is substantially faster than hD, since the latter method necessarily needs to launch m independent computations for single points $\boldsymbol{z}_j$, $j=1,\ldots,m$. In particular, for algorithm `C2(M)` (and `R2(M)`) we have the remarkable fact that the depths of all points in the dataset can be computed with the same complexity as a single point $\boldsymbol{z}$.

Comparison with Dyckerhoff and Mozharovskyi (2016)

We conclude this section with a detailed analysis of the previously mentioned similarities and differences between the computation of the angular and the conventional halfspace depth. The relationship between these concepts becomes particularly evident when contrasting the definition of the conventional hD (Equation (1)) for the special case $\boldsymbol{z}=\boldsymbol{0}$, with that of ahD (Equation (2)) for a general point $\boldsymbol{z}\in\mathbb{S}^{d-1}$:

$$hD(\boldsymbol{0}|\boldsymbol{X}) = \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}} \#\left\{i:\ \boldsymbol{p}'\boldsymbol{x}_i\geq 0\right\},$$

$$ahD(\boldsymbol{z}|\boldsymbol{X}) = \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}:\ \boldsymbol{p}'\boldsymbol{z}\geq 0} \#\left\{i:\ \boldsymbol{p}'\boldsymbol{x}_i\geq 0\right\}.$$

In both computations, the same family of halfspaces is considered. The only distinction is that, for the conventional hD, all halfspaces whose boundaries pass through the origin are relevant, whereas for the ahD, only those halfspaces that contain $\boldsymbol{z}$ are taken into account. Since $\boldsymbol{z}\in\mathbb{S}^{d-1}$ is contained in at least one of the halfspaces $H_{\boldsymbol{0},\boldsymbol{p}}$ and $H_{\boldsymbol{0},-\boldsymbol{p}}$ for any $\boldsymbol{p}\in\mathbb{S}^{d-1}$, it follows from a theoretical perspective that optimal algorithms for both problems should have the same time complexity. This is consistent with the fact that the best algorithms presented in Dyckerhoff and Mozharovskyi (2016) have complexity $\mathcal{O}(n^{d-1}\log n)$ for computing the depth of a single point in $\mathbb{R}^d$, and thus exhibit the same complexity as our algorithms `C2` and `C3`.

We now consider the scenario in which the depths of m points $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_m$ need to be computed. To this end, we take a closer look at the best algorithms for the two depth notions as presented in the respective articles: (i) The algorithms in Dyckerhoff and Mozharovskyi (2016) address the special case where the depth hD of the origin $\boldsymbol{0}\in\mathbb{R}^d$ w.r.t. a data cloud $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$ is computed. The general case—computing the depth of a point $\boldsymbol{z}_j\neq\boldsymbol{0}$ w.r.t. a data cloud—is reduced to this special case by means of the relation

$$hD\left(\boldsymbol{z}_j|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\right) = hD\left(\boldsymbol{0}|\boldsymbol{x}_1-\boldsymbol{z}_j,\ldots,\boldsymbol{x}_n-\boldsymbol{z}_j\right).$$

However, this implies that for each point $\boldsymbol{z}_j$, the data cloud w.r.t. which the depth is computed is different. Since the main step in all algorithms involves iterating over all k-element subsets I of the data cloud and projecting the points onto the orthogonal complement $S_I^\perp$, this procedure must be performed separately for each $\boldsymbol{z}_j$. Consequently, the overall complexity of computing the conventional hD for m points is m times the complexity for a single point, that is, $\mathcal{O}(mn^{d-1}\log n)$. (ii) In contrast, for the ahD, the data cloud w.r.t. which the depth is computed is the same for all points $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_m$, namely it consists of the points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$. This allows the projection step—iterating over all k-element subsets I of the data cloud—to be performed simultaneously for all $\boldsymbol{z}_j$. As shown above, this leads to a lower overall complexity for algorithms `C2(S)`, `C3`, and especially for `C2(M)`.

It is important to note that this is not a shortcoming of the algorithms in Dyckerhoff and Mozharovskyi (2016), but rather a fundamental distinction between ahD and hD. While for ahD, the same set of halfspaces $\{H_{\boldsymbol{0},\boldsymbol{p}}:\ \boldsymbol{p}\in\mathbb{S}^{d-1}\}$ is considered for all points $\boldsymbol{z}_j$, the situation is different for

Table 1 Order of time complexities of the combinatorial algorithms for computing the depth of a single point (column 1), the depths of m points simultaneously (column 2) or the depths of all points in the dataset (column 3). In each situation, we compute the depth w.r.t. a dataset $\boldsymbol{X}$ in $\mathbb{S}^{d-1}$ of size n

Algorithms	$ahD(\boldsymbol{z}\|\boldsymbol{X})$	$ahD(\boldsymbol{z}_j\|\boldsymbol{X})$, $j=1,\ldots,m$,	$ahD(\boldsymbol{x}_i\|\boldsymbol{X})$, $i=1,\ldots,n$
`C1`	$\mathcal{O}(n^d)$	$\mathcal{O}(n^d + mn^{d-1})$	$\mathcal{O}(n^d)$
`C2(S)`, `C3`	$\mathcal{O}(n^{d-1}\log n)$	$\mathcal{O}(n^{d-1}\log n + mn^{d-1})$	$\mathcal{O}(n^d)$
`C2(M)`	$\mathcal{O}(n^{d-1}\log n)$	$\mathcal{O}(n^{d-1}\log n + mn^{d-2}\log n)$	$\mathcal{O}(n^{d-1}\log n)$

the conventional hD. There, for each point $\boldsymbol{z}_j$, a different set of halfspaces $\{H_{\boldsymbol{z}_j,\boldsymbol{p}}\colon \boldsymbol{p}\in\mathbb{S}^{d-1}\}$ must be taken into account.

In Section 4.4 we provide a small simulation study comparing the computational performances of ahD and hD to illustrate the difference between the two depths.

4 Implementation and simulations

All our algorithms from Sections 2 and 3 were implemented in C++. To achieve maximum portability, no external libraries were used. The C++ codes have been interfaced to R to make them callable in the R environment. We will incorporate the algorithms into the R package **ddalpha** (Pokotylo et al. 2024) in the near future. A small R package containing only our routines for ahD and including the C++ source codes can be downloaded from GitHub (https://github.com/DyckerhoffRainer/sphericalDepth).

We have seen in Table 1 that the complexity of our computation algorithms for the angular halfspace depth in $\mathbb{S}^{d-1}$ is comparable with that of the (Euclidean) halfspace depth in $\mathbb{R}^d$. As discussed in Section 2.5, one major difference is that our algorithms should give a significant time reduction when the depths of several points $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_m$ have to be computed w.r.t. the same dataset. Therefore, we implemented the algorithms in such a way that not only a single point $\boldsymbol{z}$ could be passed to the routine but an arbitrary number m of data points $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_m$ can be provided. When computing in-sample depth values—that is, when $\boldsymbol{z}_j$ coincide with some data points $\boldsymbol{x}_i$—an additional simplification arises. Specifically, during the projection step, the projections of these points need not be computed twice (once as part of $\boldsymbol{x}_i$ and again as part of $\boldsymbol{z}_j$). Consequently, an extra parameter can be passed to our routines containing the indices of sample points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$ for which the angular halfspace depth must be calculated. Note, however, that these optimizations do not alter the computational complexity of the routines.

Our algorithms can be easily generalized in such a way that not only the depth w.r.t. an empirical distribution can be computed but also w.r.t. a general discrete distribution with finite support on $\mathbb{S}^{d-1}$. In our codes we already implemented this generalization by allowing the user to specify (integer or real) weights for the data points.

To compare the proposed variants of the algorithm and compare them with the routine `sdepth` from the R package **depth** (Genest et al. 2019), we conducted a large number of simulations. All simulations were run on a single core of an Intel® Core™ i5-4670 CPU. We now shortly describe the different experiments that were conducted as well as the main findings.

4.1 Comparison of exact algorithms: Depth of a single point

First, we compare the different variants of our algorithm in the case of data in general position, and a single point $\boldsymbol{z}$. We simulated datasets from the uniform distribution on the unit sphere for dimensions $d=3,\ldots,10$. We choose the sample sizes $n=10\cdot 2^i$, $i=2,\ldots,13$, i.e., up to $n=81\,920$. The point $\boldsymbol{z}$ was also sampled from the uniform distribution on the sphere. For each choice of the parameters, $n_{sim}=10$ (or $n_{sim}=100$ if the computation time was short) independent runs were conducted. The computation times were averaged over these n_{sim} simulations. The average execution times in seconds are shown in Table 2. All times are reported with three significant digits. Computations with a single run taking more than one hour, i.e., above 3 600 seconds, were halted. For each combination of d and n the times of the compared algorithms are marked on a color scale which ranges from green (low values) to red (large values). The following observations can be derived from Table 2:

- As expected, the variants `R1` and `C1` are clearly inferior because of their worse complexity; see also Table 1.
- In dimension $d=3$ there is no difference between `R3` and `C3` since there is no dimension reduction used, and both routines simply call the exact algorithm for $d=3$ from Section 2.4. Therefore, all the differences for these two methods in Table 2 are just due to randomness. The same holds for `R2` and `C2` since in both algorithms the dimension is reduced from $d=3$ to $d=2$. This observation is confirmed by the execution times in Table 2.
- In dimension $d=3$ the best algorithms are `C3(GP)` (and `R3(GP)`) which do assume that the points are in general position. From the universal algorithms `C2(S)` and `C3(NGP)` (and of course also `R2(S)` and `R3(NGP)`)

are comparable with advantages for `C3(NGP)` when n is small and for `C2(S)` when n is large. However the variants `C2(M)` and `R2(M)` are not much worse.
- For the same reason as above, in $d = 4$ there should be no significant difference between `R3` and `C3`. This is again confirmed by the execution times in Table 2. However, the `C2` variants are clearly faster than the corresponding `R2` variants. The winner here is `C2(S)`. Only for small values of n, `C3(GP)` (and `R3(GP)`) are faster.
- For $d > 4$ the recursive variants of the algorithm are clearly inferior. Therefore, they were no longer used when $d > 7$. The clear winner in dimension $d > 4$ is `C2(S)` followed by `C2(M)`.

Therefore, `C2(S)` can be declared the overall winner. Only for $d = 3$ the variants `C3(GP)` and `R3(GP)` are slightly better. Note also, that `C2(M)` is never much worse than `C2(S)`. For the parameter combination $d = 3, n = 81\,920$, the algorithms `C3` and `R3` failed because of insufficient main memory. This is because an array with $\binom{81\,921}{2}$ entries and a total size of more than 50 GB could not be allocated in **Step 3** of the algorithm. This is marked by a star in Table 2.

4.2 Comparison of exact algorithms: Depth of multiple points

We identified the four algorithms `C2(S)`, `C2(M)`, `C3(NGP)`, and `C3(GP)` as the most promising candidates. We now compare those four algorithms in the case where the angular halfspace depth of a large number of points has to be computed w.r.t. the same dataset. We used the same setup as in Section 4.1, but instead of a single point z, the depth of $m = 1\,000$ points $z_1, \ldots, z_{1000}$ is computed. The points z_j are chosen from the uniform distribution on $\mathbb{S}^{d-1}$. Table 3 presents the results of our study. Here, the clear winner is algorithm `C2(M)` which performs better than `C2(S)`, `C3(GP)`, and `C3(NGP)` in all the considered situations. The speedup factor of simultaneously computing the 1 000 depths compared to calling the respective algorithm 1 000 times with only a single point z_j lies between 61.6 and 976 for `C2(M)`, between 38 and 146 for `C2(S)`, between 7.19 and 18 for `C3(NGP)` and between 8.31 and 19.7 for `C3(GP)`. As expected, the speedup factor increases with growing n.

In another simulation, we considered computing the depths of all sample points w.r.t. the sample itself. Again, we used the same setup as in Section 4.1, but instead of a single point z, the depth of the points $x_1, \ldots, x_n$ is computed. Table 4 presents the results of our study. Again, the clear winner is algorithm `C2(M)`, which performs better than the other three considered variants in all the considered situations with the only exception for $d = 3, n = 40$. The ratio of the time of simultaneously computing the depths of all sample points and the time for calling the respective algorithm for a single point does never exceed 1.24 (for the combination $d = 3, n = 20\,480$) for algorithm `C2(M)`. This means that computing the depths of all sample points takes only 24% more time than computing the depth of a single sample point. Contrasting that with the ratios for the other variants of the algorithm, we choose the combination $d = 3, n = 5\,120$. Here, the ratios are 1.09 for `C2(M)`, 30.9 for `C2(S)`, 169 for `C3(NGP)` and 172 for `C3(GP)`.

These results show, that in the case where the depths of several points have to be computed, the clear winner is `C2(M)`.

4.3 Comparison with `sdepth`

In the previous sections we identified `C2(S)` as the best performing algorithm when the depth of a single depth has to be computed, and `C2(M)` as the best algorithm when the depths of multiple points have to be computed. Therefore, we compared `C2(S)` and `C2(M)` with the procedure `sdepth` from the R package **depth** (Genest et al. 2019). In that package, `sdepth` is implemented in pure R code, without interfacing to C++ or some other compiled language. Since R codes are generally known to be much slower than C++ routines, it is quite clear that comparing our C++ implementation with the R code will not be fair. Hence, we implemented `sdepth` also in C++ to conduct an impartial comparison. Since `sdepth` only works for $d \leq 3$, we compared `C2(S)` and `C2(M)` with `sdepth` only in dimensions $d = 2, 3$. We performed comparisons (i) for the case where only the depth of a single point has to be computed (Section 4.1), (ii) for the case when the depths of 1 000 points have to be computed simultaneously w.r.t. the same dataset (Section 4.2), and also (iii) for the case that the depths of all sample points have to be computed. The results of these comparisons are shown in Table 5 (for $d = 2$) and 6 (for $d = 3$). While in $d = 2$ and the case that the depth of a single point has to be computed both algorithms `C2(S)` and `sdepth` have nearly identical running times, in all other situations, `C2(S)` or `C2(M)` is the clear winner and beats `sdepth` by up to five orders of magnitude.

4.4 Comparison with Euclidean halfspace depth

To illustrate the differences between the computation of the ahD and the Euclidean hD discussed at the end of Section 3, we conducted a small simulation study. For the sake of brevity, all comparisons were performed in dimension $d = 3$. Following previous analyses, we considered three scenarios: (i) computing the depth of a single point (Task 1), (ii) of $m = 1\,000$ points (Task 2), and (iii) of all sample points (Task 3).

For the ahD, algorithm `C2(M)` was employed in all three cases, as it outperformed other approaches in most of our comparisons. Similarly, for the conventional hD, we selected the combinatorial algorithm with $k = d-2$ from Dyckerhoff

Table 2 Computation times for computing the angular halfspace depth of a single point z w.r.t. a sample of size n from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$. All points are in general position. Times are averaged over at least 10 independent runs. They are given in seconds with three significant digits

		n											
d	Algorithm	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	R1	0.00126	0.00928	0.0746	0.595	4.64	36.6	291					
3	R2(s)	0.00017	0.00053	0.00223	0.00893	0.036	0.149	0.628	2.68	11.2	46.6	213	865
3	R2(m)	0.0002	0.00068	0.00264	0.0104	0.0416	0.173	0.719	3.06	12.5	52.5	241	963
3	R3(nGP)	0.00016	0.00049	0.00221	0.00849	0.0359	0.155	0.667	2.86	12.3	53.2	229	*
3	R3(GP)	0.00008	0.00027	0.00111	0.005	0.0209	0.091	0.395	1.71	7.47	33.4	145	*
3	C1	0.001	0.00568	0.0359	0.247	1.78	14.1	113					
3	C2(s)	0.00014	0.00054	0.00231	0.00909	0.036	0.148	0.612	2.57	10.9	45	190	833
3	C2(m)	0.00026	0.00074	0.0028	0.0108	0.0417	0.172	0.709	2.95	12.2	51	234	930
3	C3(nGP)	0.00011	0.00048	0.00223	0.00907	0.0369	0.157	0.672	2.88	12.3	53.2	228	*
3	C3(GP)	0.0001	0.00035	0.00135	0.00542	0.0221	0.0932	0.4	1.72	7.49	33.3	144	*
4	R1	0.047	0.725	11.5	183								
4	R2(s)	0.00609	0.0426	0.331	2.81	22.8	189						
4	R2(m)	0.00689	0.0507	0.391	3.23	26.5	220						
4	R3(nGP)	0.00414	0.0341	0.295	2.57	22.9	198						
4	R3(GP)	0.00229	0.0191	0.167	1.47	13.3	117	1020					
4	C1	0.0149	0.172	2.11	29.7	449							
4	C2(s)	0.00309	0.0227	0.171	1.39	11.7	96.9	805					
4	C2(m)	0.00382	0.0265	0.201	1.64	13.6	112	927					
4	C3(nGP)	0.00412	0.0346	0.296	2.57	22.9	198						
4	C3(GP)	0.00233	0.0196	0.168	1.46	13.3	117	1020					
5	R1	1.75	54.5	1780									
5	R2(s)	0.189	2.96	49.9	854								
5	R2(m)	0.257	3.57	59.5	1010								
5	R3(nGP)	0.166	2.61	46.1	815								
5	R3(GP)	0.0886	1.46	26.1	462								
5	C1	0.174	3.86	99	2750								
5	C2(s)	0.0429	0.611	9.61	156	2570							
5	C2(m)	0.0508	0.709	11.2	181	2980							
5	C3(nGP)	0.0782	1.31	23.3	407								
5	C3(GP)	0.0418	0.736	13.1	231								
6	R1	58.4											
6	R2(s)	6.5	227										
6	R2(m)	7.81	272										
6	R3(nGP)	5.24	198										
6	R3(GP)	2.89	111										
6	C1	1.42	69										
6	C2(s)	0.43	13	411									
6	C2(m)	0.485	14.9	473									
6	C3(nGP)	0.888	33.1	1200									
6	C3(GP)	0.5	18.7	682									
7	R1	2000											
7	R2(s)	230											
7	R2(m)	277											
7	R3(nGP)	183											
7	R3(GP)	101											
7	C1	10.1	1020										
7	C2(s)	3.41	211										
7	C2(m)	3.8	239										
7	C3(nGP)	7.89	623										
7	C3(GP)	4.44	350										
8	C1	58.6											
8	C2(s)	21.8	2780										
8	C2(m)	24	3140										
8	C3(nGP)	53.9											
8	C3(GP)	30.7											
9	C1	291											
9	C2(s)	119											
9	C2(m)	130											
9	C3(nGP)	301											
9	C3(GP)	173											
10	C1	1240											
10	C2(s)	537											
10	C2(m)	579											
10	C3(nGP)	1390											
10	C3(GP)	812											

Table 3 Computation times for computing the angular halfspace depth of $m = 1\,000$ points $z_1, \ldots, z_{1000}$ w.r.t. a sample of size n from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

d	Algorithm	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	C2(s)	0.00368	0.00923	0.0264	0.0862	0.293	1.08	4.36	17.7	74.3	339	1570	
3	C2(m)	0.00306	0.00676	0.0161	0.0403	0.105	0.303	0.995	3.56	13.6	54.6	239	984
3	C3(nGP)	0.0148	0.0444	0.154	0.621	2.23	8.92	38.8	163	687			
3	C3(GP)	0.00734	0.0227	0.078	0.296	1.12	5.22	22.3	99.2	494			
4	C2(s)	0.0732	0.369	2.15	13.7	93.5	697						
4	C2(m)	0.062	0.279	1.49	6.36	33.6	198	1300					
4	C3(nGP)	0.553	3.72	25.8	183	1400							
4	C3(GP)	0.276	1.79	12.5	91.5	716							
5	C2(s)	0.9	9.83	112	1450								
5	C2(m)	0.783	7.26	70.5	699								
5	C3(nGP)	10.1	139	2000									
5	C3(GP)	4.96	67.6	966									
6	C2(s)	9	197										
6	C2(m)	7.74	150	2960									
6	C3(nGP)	120	3400										
6	C3(GP)	60.1	1720										
7	C2(s)	64.7	3020										
7	C2(m)	57.2	2350										
7	C3(nGP)	1100											
7	C3(GP)	534											
8	C2(s)	377											
8	C2(m)	336											
8	C3(nGP)												
8	C3(GP)												
9	C2(s)	1900											
9	C2(m)	1700											
9	C3(nGP)												
9	C3(GP)												

Table 4 Computation times for computing the angular halfspace depth of all n data points in a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$ w.r.t. the sample itself. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

d	Algorithm	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	C2(s)	0.0003	0.00112	0.00522	0.0305	0.188	1.31	10	79.5	657			
3	C2(m)	0.00022	0.00077	0.00306	0.0127	0.0509	0.191	0.776	3.22	13.5	63.1	258	1120
3	C3(nGP)	0.00038	0.00226	0.0153	0.144	0.923	7.03	57.8	485				
3	C3(GP)	0.0002	0.00142	0.00811	0.069	0.455	3.67	30.7	297				
4	C2(s)	0.00453	0.0398	0.399	4.76	58.8	820						
4	C2(m)	0.00383	0.0305	0.229	1.81	14.8	122	1010					
4	C3(nGP)	0.0117	0.165	2.45	35.7	563							
4	C3(GP)	0.00666	0.0897	1.38	18.3	281							
5	C2(s)	0.0589	1.06	21.5	491								
5	C2(m)	0.0515	0.777	12.1	201	3260							
5	C3(nGP)	0.213	6.33	186									
5	C3(GP)	0.119	3.35	96.7									
6	C2(s)	0.585	21.5	871									
6	C2(m)	0.53	16.1	511									
6	C3(nGP)	2.67	154										
6	C3(GP)	1.41	84.5										
7	C2(s)	4.67	351										
7	C2(m)	4.07	274										
7	C3(nGP)	21.9	2850										
7	C3(GP)	12.2	1570										
8	C2(s)	28											
8	C2(m)	25.5	3370										
8	C3(nGP)	144											
8	C3(GP)	82											
9	C2(s)	148											
9	C2(m)	138											
9	C3(nGP)	780											
9	C3(GP)	448											
10	C2(s)	654											
10	C2(m)	613											
10	C3(nGP)												
10	C3(GP)	2040											

Table 5 Computation times for computing the angular halfspace depth of a single point z (Task 1), for $m = 1\,000$ points $z_1, \ldots, z_m$ (Task 2) or all data points $x_1, \ldots, x_n$ in the sample (Task 3) w.r.t. a sample of size n from a uniform distribution on the 1-sphere $\mathbb{S}^1$. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

$d = 2$	Task 1: single point		Task 2: $m = 1\,000$ points		Task 3: all data points	
n	C2(s)	sdepth	C2(m)	sdepth	C2(m)	sdepth
10^2	0.000018	0.000018	0.000091	0.00725	0.000018	0.00073
10^3	0.000110	0.000162	0.000270	0.102	0.000140	0.107
10^4	0.00137	0.00161	0.00174	1.25	0.00190	12.4
10^5	0.0162	0.0186	0.0187	15.7	0.0200	1550
10^6	0.224	0.221	0.236	185	0.255	
10^7	3.22	3.34	3.26		3.80	

Table 6 Computation times for computing the angular halfspace depth of a single point z (Task 1), for $m = 1\,000$ points $z_1, \ldots, z_m$ (Task 2) or all data points $x_1, \ldots, x_n$ in the sample (Task 3) w.r.t. a sample of size n from a uniform distribution on the 2-sphere $\mathbb{S}^2$. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

$d = 3$	Task 1: single point		Task 2: $m = 1\,000$ points		Task 3: all data points	
n	C2(s)	sdepth	$C2(m)$	sdepth	$C2(m)$	sdepth
40	0.00014	0.0003	0.00306	0.25	0.00022	0.0103
80	0.00054	0.00182	0.00676	1.88	0.00077	0.143
160	0.00231	0.0136	0.0161	13.6	0.00306	2.35
320	0.00909	0.111	0.0403	111	0.0127	35.6
640	0.036	1.08	0.105	995	0.0509	637
1280	0.148	8.9	0.303	8690	0.191	11100
2560	0.612	72.1	0.995		0.776	
5120	2.57	583	3.56		3.22	
10240	10.9		13.6		13.5	
20480	45		54.6		63.1	
40960	190		239		258	
81920	833		984		1120	

and Mozharovskyi (2016) based on its superior performance in most scenarios.

The results of these comparisons are summarized in Table 7. While both algorithms exhibit similar performance for Task 1, significant differences emerge when computing depths for multiple points w.r.t. the same data cloud (Tasks 2 and 3). For a data set with $n = 5\,120$ points, the ahD of 1 000 additional points (Task 2) or of the entire sample (Task 3) is computed in approximately 3.5 seconds. In contrast, the conventional hD requires more than 40 minutes for Task 2 and nearly 3.5 hours for Task 3.

4.5 Depth of points without general position

In our final numerical study, we compare the variants of our algorithm in the case that the points are not in a general position. For that, we use the same setup as in the first experiment (Section 4.1, with results shown in Table 2). The only difference is that the dataset is not simulated from the uniform distribution on the sphere, but from the discrete distribution $\boldsymbol{X}$ given by

$$\boldsymbol{X} = U \frac{(\boldsymbol{Z}, 1)'}{\|(\boldsymbol{Z}, 1)'\|},$$

where $U \sim \mathcal{U}(\{-1, +1\})$ is uniform on $\{-1, +1\}$, and $\boldsymbol{Z} = (Z_1, \ldots, Z_{d-1})'$ is given by

$$Z_i = \tan\left(\frac{Y_i}{N} \cdot \frac{\pi}{2}\right), \quad i = 1, \ldots, d-1,$$

where $Y_i \sim \mathcal{U}(\{-(N-1), -(N-3), \ldots, N-3, N-1\}), i = 1, \ldots, d-1$. Here, $\mathcal{U}(A)$ is the uniform distribution on a set A. The random variables $U, Y_1, \ldots, Y_{d-1}$ are independent.

This yields a uniform distribution on a grid of N^{d-1} points on the upper hemisphere $\mathbb{S}_+^{d-1}$ and the same number of points on the lower hemisphere $\mathbb{S}_-^{d-1}$. The grid is formed by the intersection of $(d-1)N^{d-2}$ great circles (without the intersection points on the equator). The number N of subdivisions is chosen such that $N \geq 4$ and that there are at least as many grid points as sample points, i.e. $N = \max\left\{\lceil \sqrt[d-1]{\frac{n}{2}} \rceil, 4\right\}$. The grid is illustrated in Figures 5 and 6. The point z, whose depth is to be computed, is still chosen from the uniform distribution on $\mathbb{S}^{d-1}$. The results are shown in Table 8.

Compared to the data in general position, the results are slightly different, but the big picture is very similar. As in the case of a uniform distribution, variants `R1` and `C1` are inferior because of their worse complexity. For $d = 3$ there is no difference between `R3` and `C3`, as well as between `R2` and

Table 7 Computation times for computing the angular and the conventional halfspace depth of a single point z (Task 1), for $m = 1\,000$ points $z_1, \ldots, z_m$ (Task 2) or all data points $x_1, \ldots, x_n$ in the sample (Task 3) w.r.t. a sample of size n from a uniform distribution on the 2-sphere $\mathbb{S}^2$ (angular halfspace depth) and a standard trivariate normal distribution (conventional halfspace depth). All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

$d = 3$	Task 1: single point		Task 2: $m = 1\,000$ points		Task 3: all data points	
n	$\overline{C2}$(m)	hD	$\overline{C2}$(m)	hD	$\overline{C2}$(m)	hD
40	0.00026	0.00016	0.00306	0.114	0.00022	0.00895
80	0.00074	0.00056	0.00676	0.479	0.00077	0.0508
160	0.0028	0.00232	0.0161	2.06	0.00306	0.444
320	0.0108	0.0118	0.0403	8.64	0.0127	3.6
640	0.0417	0.0403	0.105	34.6	0.0509	27.3
1280	0.172	0.196	0.303	152	0.191	224
2560	0.709	0.728	0.995	643	0.776	1530
5120	2.95	3.41	3.56	2550	3.22	12800

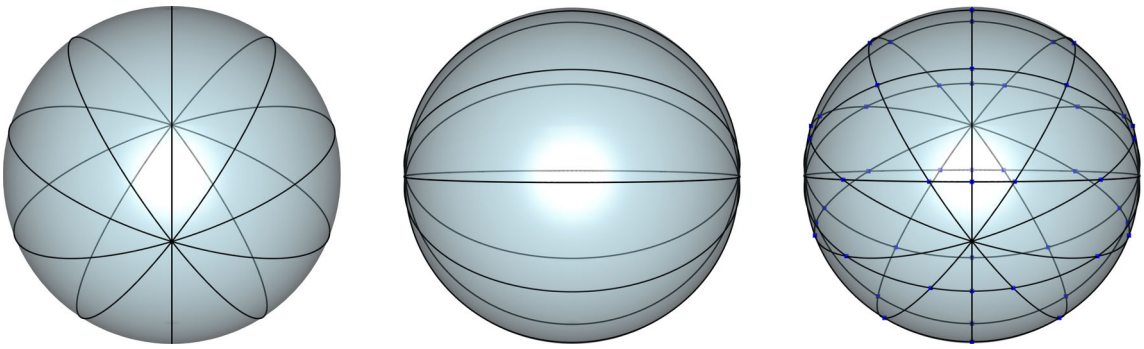


Fig. 5 Illustration of the grid used in the simulations in Section 4.5 for $d = 3$, $N = 5$. The left-hand and middle pictures each show five great circles. In the right-hand picture, all ten great circles are shown. The 50 points that constitute the grid are shown in blue in the right-hand picture

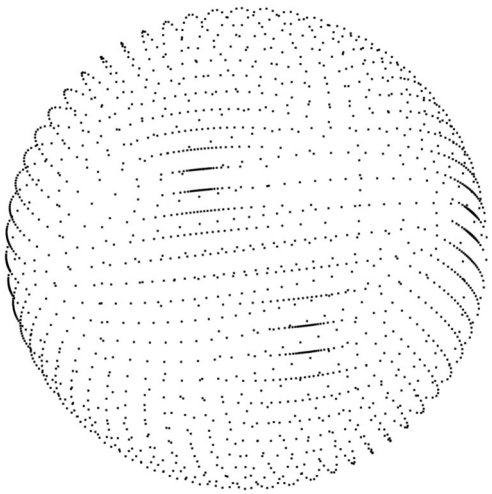


Fig. 6 Illustration of the grid of the support points of $\boldsymbol{X}$ in Section 4.5 for $d = 3$, $N = 30$

`C2`; for $d = 4$ there is no significant difference between `R3` and `C3`. For $d > 4$ the recursive variants of the algorithm are inferior and are no longer used when $d > 7$.

In dimensions $d = 3, 4$, the variants `C3(NGP)` and `R3(NGP)` perform the best, followed by `C2(M)` and `C2(S)`. In $d = 5$ the best algorithms are `C2(S)`, `C2(M)` and `C3(NGP)`, which perform very similar, although `C2(M)` is always slower than `C2(S)`. For $d > 5$ the clear winner is `C2(S)` followed closely by `C2(M)` and `C3(NGP)` on the third place.

As in the simulations in Section 4.2 we also consider the case that the depths of $m = 1\,000$ points w.r.t. the same dataset have to be computed, as well as the case when the depths of all points in a sample have to be computed. The corresponding results are given in Tables 9 and 10. In these simulations, we only applied the three algorithms `C2(S)`, `C2(M)` and `C3(NGP)` since they proved to have the best performance. The picture here is very clear: In Tables 9 and 10, the best algorithm is always `C2(M)` followed by `C2(S)` on the second, and `C3(NGP)` on the third place.

4.6 Parallelization of the algorithms

The proposed algorithms can be easily parallelized. In the main loop of the algorithms, for all k-element subsets of data points that appear in (17) in Theorem 5, the angular halfspace depth w.r.t. a suitably projected dataset must be computed.

Table 8 Computation times for computing the angular halfspace depth of a single point z w.r.t. a sample of size n from a discrete distribution on $\mathbb{S}^{d-1}$. The sample will usually not be in a general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

d	Algorithm	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	R1	0.00115	0.00931	0.0751	0.587	4.81	39.9	326					
3	R2(s)	0.00015	0.00059	0.00221	0.00878	0.0373	0.157	0.664	2.79	11.7	49	208	908
3	R2(m)	0.0002	0.00052	0.00245	0.0096	0.0397	0.167	0.699	2.93	12.3	51.5	218	949
3	R3(nGP)	0.00005	0.00021	0.0005	0.0019	0.0071	0.0309	0.12	0.511	2.18	9.29	39.6	
3	C1	0.00077	0.00336	0.0151	0.0865	0.53	4.07	29.3					
3	C2(s)	0.00013	0.00046	0.00133	0.00443	0.0159	0.0657	0.254	1.06	4.41	18.4	76.8	322
3	C2(m)	0.00017	0.00049	0.00138	0.00483	0.0176	0.0717	0.279	1.16	4.82	20.1	93.2	370
3	C3(nGP)	0.00011	0.00025	0.0008	0.00241	0.0083	0.0332	0.126	0.522	2.2	9.32	39.7	
4	R1	0.0435	0.686	11.3	184								
4	R2(s)	0.00495	0.0385	0.327	2.77	23.2	198						
4	R2(m)	0.00535	0.041	0.347	2.94	24.4	208						
4	R3(nGP)	0.00261	0.0118	0.0879	0.652	3.85	36.7						
4	C1	0.0424	0.389	4.42	43.2	435							
4	C2(s)	0.00253	0.0115	0.0811	0.535	3.33	29.7	219	1910				
4	C2(m)	0.00274	0.012	0.0866	0.572	3.59	32	238	2080				
4	C3(nGP)	0.00221	0.00862	0.0634	0.446	2.44	24.3	188	1800				
5	R1	1.63	53.9	1760									
5	R2(s)	0.19	3.13	52.1	852								
5	R2(m)	0.215	3.44	55.6	887								
5	R3(nGP)	0.132	1.72	18.8	161								
5	C1	1.49	79.9	4000									
5	C2(s)	0.0516	0.731	7.29	63.9	1150							
5	C2(m)	0.0552	0.771	7.67	65.7	1220							
5	C3(nGP)	0.0629	0.758	6.96	43.9	1180							
6	R1	59.6											
6	R2(s)	7.08	244										
6	R2(m)	8.2	276										
6	R3(nGP)	5.64	183										
6	C1	24.4											
6	C2(s)	0.587	17.1	517									
6	C2(m)	0.631	18.3	549									
6	C3(nGP)	0.949	29.4	819									
7	R2(s)	251											
7	R2(m)	292											
7	R3(nGP)	211											
7	C2(s)	4.72	343										
7	C2(m)	5.07	379										
7	C3(nGP)	8.95	616										
8	C2(s)	29.5	4480										
8	C2(m)	31.7	4850										
8	C3(nGP)	62.2	9980										

Table 9 Computation times for computing the angular halfspace depth of $m = 1\,000$ points $z_1, \ldots, z_{1000}$ w.r.t. a sample of size n from a discrete distribution on $\mathbb{S}^{d-1}$. The sample will usually not be in a general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

d	Algorithm	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	C2(s)	0.00229	0.0057	0.0111	0.0325	0.101	0.358	1.26	5.14	21.1	88.7	411	1860
3	C2(m)	0.0021	0.00505	0.0094	0.0231	0.0573	0.157	0.456	1.54	5.64	22	87.7	382
3	C3(nGP)	0.00511	0.0171	0.0377	0.139	0.495	1.96	7.18	30.4	122	512	2510	
4	C2(s)	0.0459	0.147	0.733	3.71	17.9	134	974					
4	C2(m)	0.0438	0.139	0.632	2.75	11.6	68.7	384	2710				
4	C3(nGP)	0.262	0.851	5.38	34.4	171	1500						
5	C2(s)	0.784	6.79	49.3	312	4740							
5	C2(m)	0.735	6.08	42	258	3140							
5	C3(nGP)	7.58	73.1	565	3370								
6	C2(s)	8.79	178	3450									
6	C2(m)	8	151	2700									
6	C3(nGP)	109	2800										
7	C2(s)	66.3	3120										
7	C2(m)	60.1	2550										
7	C3(nGP)	1060											
8	C2(s)	395											
8	C2(m)	359											
8	C3(nGP)	7620											

Table 10 Computation times for computing the angular halfspace depth of all n data points in a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ from a discrete distribution on the unit sphere $\mathbb{S}^{d-1}$ w.r.t. the sample itself. The sample will usually not be in a general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits

d	Algorithm	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	C2(s)	0.000201	0.000652	0.0023	0.0127	0.0669	0.396	2.65	21.2	173	1450		
3	C2(m)	0.000174	0.000566	0.00163	0.00707	0.0266	0.0878	0.34	1.41	5.84	24.5	114	487
3	C3(nGP)	0.000152	0.000719	0.00334	0.0302	0.215	1.51	11.3	91.7	747	6290		
4	C2(s)	0.00346	0.0172	0.144	1.25	10.5	143	2020					
4	C2(m)	0.00311	0.014	0.102	0.704	4.5	38.9	292	2480				
4	C3(nGP)	0.00508	0.0299	0.428	5.93	60	1060						
5	C2(s)	0.0705	1.08	11.1	109	3070							
5	C2(m)	0.0643	0.933	8.42	74.4	1510							
5	C3(nGP)	0.15	2.72	46.5	537								
6	C2(s)	0.771	24.9	773									
6	C2(m)	0.706	20.5	581									
6	C3(nGP)	2.3	116	5330									
7	C2(s)	7.39	415										
7	C2(m)	6.9	346										
7	C3(nGP)	21.7	2650										
8	C2(s)	34.8											
8	C2(m)	32.9											
8	C3(nGP)	152											

Since the computations of the angular halfspace depths are completely independent of each other, they can be executed in parallel. To investigate the effects of parallelization we used the OpenMP® API to implement a parallel version of the combinatorial algorithms (as they proved to be superior to the recursive algorithms). Only the main loop was parallelized in this version of the algorithm. In the original algorithm, the generation of all k-element subsets was accomplished using a successor algorithm, that is, in each iteration the current subset was constructed from the subset in the previous iteration. Using such a successor algorithm clearly prevents the parallelization of the main loop. Therefore, the successor algorithm has to be replaced by an algorithm where each k-element subset is generated independently of the other subsets. Such algorithms are commonly known as *unranking algorithms* and are described, e.g., in Kreher and Stinson (1999, Section 2.3). In particular, we applied a variant of Algorithm 2.10 given there.

The parallel algorithm was again run on an Intel® Core™ i5-4670 CPU. This CPU has four processor cores. In the tables given in Appendix C we report the times for the algorithm running on a single core and on all four cores as well as the respective ratio. Data was generated from a uniform distribution on $\mathbb{S}^{d-1}$. We only show the results for algorithm `C2(M)` as this algorithm is the overall winner. The results for other combinatorial algorithms are, however, similar. We again considered the three tasks of computing the angular halfspace depth of a single point $\boldsymbol{z}$ (Table 11 in Appendix C), of $m = 1\,000$ points $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m$ (Table 12 in Appendix C) and of all data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in the sample (Table 13 in Appendix C). In all tables, ratios between 3 and 3.5 are marked in yellow, whereas ratios of at least 3.5 are marked in orange. Clearly, one cannot expect to get a speedup factor equal to the number of cores. However, in almost all considered scenarios the speedup when running on all four cores is greater than three. This shows that our algorithms benefit significantly from running on a multi-core system.

As the Intel® Core™ i5-4670 was introduced already in the second quarter of 2013, it is a rather old CPU. However, when studying the effect of parallelization, this CPU has the advantage that all of its four cores are identical and that one can rely (more or less) on the nominal CPU frequency whereas in modern CPUs there are different types of cores (so-called Performance-cores and Efficient-cores) that typically run with different frequencies which may additionally vary depending on the core temperatures and other factors. Clearly, this makes it difficult to estimate the effect of parallelizing the algorithm. However, to give an impression what is possible on a modern CPU we also run a small simulation on an Intel® Core™ i5-13600 CPU which is a current CPU in the mid-range segment. This CPU has 14 cores, 6 Performance-cores and 8 Efficient-cores. In this simulation, all 14 cores were used. The results of this simulation study are shown in Table 14 in Appendix C.

4.7 Simulation study: Conclusion

To sum up our findings, one can say that the overall winner is our exact algorithm `C2`. The variant `C2(S)` should be used when the depth of a single point (or a small number of points) has to be computed, variant `C2(M)` when the depths of several points have to be computed. There are only a few situations where different algorithms, such as `C3` or `R3`, perform better. Still, in these situations `C2(S)` or `C2(M)` are not much worse. Moreover, the numerical problems in the case of points not in general position are much smaller for variant `C2` compared to `C3` or `R3`. As already mentioned in Section 2.3, the gnomonic projection strongly inflates rounding errors in the data when points are close to the equator. Therefore, the common technique of treating two floating point numbers as

equal when their difference is less than some given small tolerance cannot be applied since this tolerance should depend on the values of the numbers themselves. Hence, it is difficult to decide whether two points in the space $\mathcal{G}$ of the gnomonic projection are identical or not. In the `C2` algorithms, we cope with this problem by applying the arc-tangent transform to the points in $\mathcal{G}$. We found no easy way to cope with this problem in the `C3` algorithms. Thus, the use of the `C3` variants is problematic unless it is known that the points are in general position.

Acknowledgements We would like to thank an anonymous referee whose comments helped to improve the presentation of this paper. Stanislav Nagy was supported by the Czech Science Foundation (project n. 24-10822S) and the ERC CZ grant LL2407 of the Ministry of Education, Youth and Sport of the Czech Republic.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Competing Interests The authors declare that they have no conflict of interest.

Appendix A Proof of Theorem 3

In the angular halfspace depth (2), we only consider halfspaces $H_{\mathbf{0},\boldsymbol{p}}$ whose boundary passes through the origin $\mathbf{0} \in \mathbb{R}^d$. Denote for $\boldsymbol{p} \in \mathbb{S}^{d-1}$ by $\mathcal{G}_{\boldsymbol{p}}$ the intersection of $\mathcal{G}$ and $H_{\mathbf{0},\boldsymbol{p}}$. When considered as a subset of the affine space $\mathcal{G}$, the set $\mathcal{G}_{\boldsymbol{p}}$ is what we call a *generalized halfspace*: (i) It is a halfspace in $\mathcal{G}$ if $\boldsymbol{p} \in \mathbb{S}^{d-1} \setminus \{\boldsymbol{e}_d, -\boldsymbol{e}_d\}$, (ii) it is the whole subspace $\mathcal{G}$ if $\boldsymbol{p} = \boldsymbol{e}_d$, and (iii) it equals an empty set if $\boldsymbol{p} = -\boldsymbol{e}_d$. By the definition (3) of the gnomonic projection ξ, a point $\boldsymbol{y} = (y_1, \dots, y_d)' \in \mathbb{S}^{d-1} \setminus \mathbb{S}^{d-1}_0$ lies in a halfspace $H_{\mathbf{0},\boldsymbol{p}}$ if and only if

$$\boldsymbol{y}'\boldsymbol{p} = \left(\xi(\boldsymbol{y})'\boldsymbol{p}\right) y_d \geq 0.$$

If $\boldsymbol{y} \in \mathbb{S}^{d-1}_+$, that is $y_d > 0$, the inequality above is equivalent with $\xi(\boldsymbol{y}) \in H_{\mathbf{0},\boldsymbol{p}} \cap \mathcal{G} = \mathcal{G}_{\boldsymbol{p}}$; for $\boldsymbol{y} \in \mathbb{S}^{d-1}_-$ we have that $\boldsymbol{y} \in H_{\mathbf{0},\boldsymbol{p}}$ if and only if $\xi(\boldsymbol{y})'\boldsymbol{p} \leq 0$, that is $\xi(\boldsymbol{y}) \in \mathcal{G}_{-\boldsymbol{p}}$.

We use this to express the number of points from $\boldsymbol{X}$ in any halfspace $H_{\mathbf{0},\boldsymbol{p}}$

$$\begin{aligned}
\#\{i\colon\ \boldsymbol{x}_i \in H_{\mathbf{0},\boldsymbol{p}}\} &= \#\left\{i\colon \boldsymbol{p}'\boldsymbol{x}_i \geq 0\right\} \\
&= \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_+ \text{ and } \xi(\boldsymbol{x}_i)'\boldsymbol{p} \geq 0\right\} \\
&\quad + \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_- \text{ and } \xi(\boldsymbol{x}_i)'\boldsymbol{p} \leq 0\right\} \\
&= \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_+ \text{ and } \xi(\boldsymbol{x}_i) \in \mathcal{G}_{\boldsymbol{p}}\right\} \\
&\quad + \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_- \text{ and } \xi(\boldsymbol{x}_i) \in \mathcal{G}_{-\boldsymbol{p}}\right\} \\
&= P_+(\mathcal{G}_{\boldsymbol{p}}) + P_-(\mathcal{G}_{-\boldsymbol{p}}),
\end{aligned} \tag{18}$$

where we use P_+ and P_- to decompose the signed measure $P_\pm$ from (4) into two (non-negative) integer-valued measures

$$\begin{aligned}
P_+(B) &= \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_+ \text{ and } \xi(\boldsymbol{x}_i) \in B\right\}, \\
P_-(B) &= \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_- \text{ and } \xi(\boldsymbol{x}_i) \in B\right\},
\end{aligned}$$

giving $P_+(B) - P_-(B) = P_\pm(B)$ for every Borel set $B \subseteq \mathcal{G}$. Let us further denote, for $\boldsymbol{p} \in \mathbb{S}^{d-1}$,

$$\mathcal{G}^\circ_{\boldsymbol{p}} = \left\{\boldsymbol{y} \in \mathcal{G}\colon \boldsymbol{y}'\boldsymbol{p} > 0\right\}$$

the relative interior of the halfspace $\mathcal{G}_{\boldsymbol{p}}$ inside $\mathcal{G}$. Because for any $\boldsymbol{p} \in \mathbb{S}^{d-1}$ we have

$$\#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_-\right\} = P_-(\mathcal{G}) = P_-(\mathcal{G}_{-\boldsymbol{p}}) + P_-(\mathcal{G}^\circ_{\boldsymbol{p}})$$

we can rewrite (18) to

$$\begin{aligned}
\#\left\{i\colon \boldsymbol{x}_i \in H_{\mathbf{0},\boldsymbol{p}}\right\} &= \#\left\{i\colon \boldsymbol{x}_i \in \mathbb{S}^{d-1}_-\right\} \\
&\quad + P_+(\mathcal{G}_{\boldsymbol{p}}) - P_-(\mathcal{G}^\circ_{\boldsymbol{p}}).
\end{aligned} \tag{19}$$

Since we assumed that $\boldsymbol{z} \in \mathbb{S}^{d-1}_+$, we know that $\boldsymbol{z} \in H_{\mathbf{0},\boldsymbol{p}}$ if and only if $\xi(\boldsymbol{z}) \in \mathcal{G}_{\boldsymbol{p}}$. Plugging (19) into (2) we therefore

obtain

$$\begin{aligned} ahD(\boldsymbol{z}|\boldsymbol{X}) &= \#\left\{i : \boldsymbol{x}_i \in \mathbb{S}_-^{d-1}\right\} \\ &\quad + \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}:\, \boldsymbol{z}\in H_{\boldsymbol{0},\boldsymbol{p}}} \left(P_+\left(\mathcal{G}_{\boldsymbol{p}}\right) - P_-\left(\mathcal{G}_{\boldsymbol{p}}^{\circ}\right)\right) \\ &= \#\left\{i : \boldsymbol{x}_i \in \mathbb{S}_-^{d-1}\right\} \\ &\quad + \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}:\, \xi(\boldsymbol{z})\in\mathcal{G}_{\boldsymbol{p}}} \left(P_+\left(\mathcal{G}_{\boldsymbol{p}}\right) - P_-\left(\mathcal{G}_{\boldsymbol{p}}^{\circ}\right)\right) \\ &= \#\left\{i : \boldsymbol{x}_i \in \mathbb{S}_-^{d-1}\right\} \\ &\quad + \inf_{H\in\mathcal{H}(\mathcal{G}):\, \xi(\boldsymbol{z})\in H} \left(P_+\left(H\right) - P_-\left(H^{\circ}\right)\right) \\ &= \#\left\{i : \boldsymbol{x}_i \in \mathbb{S}_-^{d-1}\right\} \\ &\quad + \inf_{H\in\mathcal{H}(\mathcal{G}):\, \xi(\boldsymbol{z})\in H} \left(P_\pm\left(H\right) + P_-\left(\partial H\right)\right). \end{aligned} \tag{20}$$

In the last two equalities, we write $\mathcal{H}(\mathcal{G})$ for the set of all $(d-1)$-dimensional halfspaces inside the hyperplane $\mathcal{G}$, and H° and ∂H for the relative interior and the relative boundary of halfspace $H \in \mathcal{H}(\mathcal{G})$, respectively. The set H° is thus an open $(d-1)$-dimensional halfspace inside $\mathcal{G}$, and ∂H is the $(d-2)$-dimensional affine space that forms the boundary of H inside $\mathcal{G}$. In (20), we used that each halfspace $\mathcal{G}_{\boldsymbol{p}} = H \in \mathcal{H}(\mathcal{G})$ (or its relative interior H°) corresponds to exactly one halfspace $H_{\boldsymbol{0},\boldsymbol{p}}$ in $\mathbb{R}^d$ for some $\boldsymbol{p} \in \mathbb{S}^{d-1}$ (or its interior). The only exceptions are the generalized halfspaces $\mathcal{G} = \mathcal{G}_{\boldsymbol{e}_d}$ and the empty set for $\boldsymbol{p} = -\boldsymbol{e}_d$ in $\mathcal{G}$, that do not have an equivalent halfspace $H_{\boldsymbol{0},\boldsymbol{p}}$ in $\mathbb{R}^d$. The last two sets, however, can be seen not to alter the infimum on the right-hand side of (20).

It remains to show that the summand with the boundary ∂H on the right-hand side of (20) does not contribute to the infimum. This is proved in the following lemma, which is stated for general signed empirical measures in $\mathbb{R}^d$.

Lemma 7 *Take any two empirical measures P_+ and P_- in $\mathbb{R}^d$ and the associated signed empirical measure $P_\pm = P_+ - P_-$ in $\mathbb{R}^d$. Then for all $\boldsymbol{z} \in \mathbb{R}^d$ we can write*

$$\inf_{H\in\mathcal{H}:\, \boldsymbol{z}\in H} \left(P_\pm\left(H\right) + P_-\left(\partial H\right)\right) = shD(\boldsymbol{z}|P_\pm), \tag{21}$$

where $\mathcal{H}$ stands for the collection of all closed halfspaces in $\mathbb{R}^d$.

Proof Take $M \in \mathbb{R}$ and consider a halfspace $H = \{\boldsymbol{y} \in \mathbb{R}^d : \boldsymbol{p}'\boldsymbol{y} \geq M\} \in \mathcal{H}$ such that $\boldsymbol{z} \in H$. If we shift the halfspace by an amount ϵ in the direction of its outer normal $-\boldsymbol{p}$, we get $H_\epsilon = \{\boldsymbol{y} \in \mathbb{R}^d : \boldsymbol{p}'\boldsymbol{y} \geq M - \epsilon\} \in \mathcal{H}$. For $\epsilon > 0$ sufficiently small, H and H_ϵ contain both $\boldsymbol{z}$ and exactly the same points of non-zero $P_\pm$-mass, whereas no points of non-zero P_--mass lie on the boundary hyperplane ∂H_ϵ. Hence,

$$\begin{aligned} P_\pm(H_\epsilon) &= P_\pm(H) \quad \text{and} \\ P_\pm(H_\epsilon) &= P_\pm(H_\epsilon) + P_-(\partial H_\epsilon) \\ &\leq P_\pm(H) + P_-(\partial H). \end{aligned} \tag{22}$$

Thus, the infima on both sides of (21) can as well be taken over all halfspaces $H \in \mathcal{H}$ that contain $\boldsymbol{z}$ but no points of non-zero P_--mass in ∂H, i.e,

$$\begin{aligned} &\inf_{H\in\mathcal{H}:\, \boldsymbol{z}\in H} \left(P_\pm\left(H\right) + P_-\left(\partial H\right)\right) \\ &\quad = \inf_{H\in\mathcal{H}:\, \boldsymbol{z}\in H, P_-(\partial H)=0} P_\pm\left(H\right) = shD(\boldsymbol{z}|P_\pm), \end{aligned}$$

the second equality following directly from (22) again. □

Using Lemma 7 together with (20) gives the desired result of Theorem 3.

Appendix B Proof of Theorem 5

We begin by introducing additional terms and notations. We also make several observations that will be useful in the main proof. Throughout the appendix, we simplify the notation for halfspaces, and instead of $H_{\boldsymbol{0},\boldsymbol{p}}$ we write $H_{\boldsymbol{p}}$ for $\boldsymbol{p} \in \mathbb{S}^{d-1}$. A vector $\boldsymbol{p} \in \mathbb{S}^{d-1}$ is called *minimizing for the dataset* $(\boldsymbol{z}, \boldsymbol{X})$ if $\boldsymbol{z} \in H_{\boldsymbol{p}}$ and

$$ahD(\boldsymbol{z}|\boldsymbol{X}) = \#\{i : \boldsymbol{p}'\boldsymbol{x}_i \geq 0\} = \#\left\{i : \boldsymbol{x}_i \in H_{\boldsymbol{p}}\right\}.$$

For $\boldsymbol{p} \in \mathbb{S}^{d-1}$ we write

$$\begin{aligned} &I_{\boldsymbol{p}}^+ = \{i : \boldsymbol{p}'\boldsymbol{x}_i > 0\}, \quad I_{\boldsymbol{p}}^0 = \{i : \boldsymbol{p}'\boldsymbol{x}_i = 0\}, \\ &I_{\boldsymbol{p}}^- = \{i : \boldsymbol{p}'\boldsymbol{x}_i < 0\}. \end{aligned}$$

The corresponding cardinalities are denoted by

$$n_{\boldsymbol{p}}^+ = \#I_{\boldsymbol{p}}^+, \quad n_{\boldsymbol{p}}^0 = \#I_{\boldsymbol{p}}^0, \quad n_{\boldsymbol{p}}^- = \#I_{\boldsymbol{p}}^-.$$

In this notation, we have $\#\left\{i : \boldsymbol{x}_i \in H_{\boldsymbol{p}}\right\} = n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0$, and naturally also

$$ahD(\boldsymbol{z}|\boldsymbol{X}) = \inf_{\boldsymbol{p}\in\mathbb{S}^{d-1}:\, \boldsymbol{z}\in H_{\boldsymbol{p}}} (n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0) = n - \sup_{\boldsymbol{p}\in\mathbb{S}^{d-1}:\, \boldsymbol{z}\in H_{\boldsymbol{p}}} n_{\boldsymbol{p}}^-.$$

For a subset I of indices $\{1, \ldots, n\}$, $\boldsymbol{X}_I$ denotes the dataset $(\boldsymbol{x}_i)_{i\in I}$ of data points from $\boldsymbol{X}$ with indices in I. The cardinality of the set of all indices of points from $\boldsymbol{X}$ in the linear hull $\text{span}\,(\boldsymbol{X}_I) = \text{span}\,(\boldsymbol{X}_{I^*})$ of $\boldsymbol{X}_I$ is denoted by $n_{I^*} = \#I^*$.

Our first auxiliary lemma shows that the condition $\boldsymbol{z} \in H_{\boldsymbol{p}}$ in the definition (2) of ahD can be replaced by $\boldsymbol{z} \in H_{\boldsymbol{p}}^\circ$.

Lemma 8 *There exists a minimizing direction $\boldsymbol{p} \in \mathbb{S}^{d-1}$ for $(\boldsymbol{z}, \boldsymbol{X})$ such that $I^0_{\boldsymbol{p}} = \emptyset$ and $\boldsymbol{z} \in H^\circ_{\boldsymbol{p}}$, i.e., neither $\boldsymbol{z}$ nor any of the data points lie on the boundary of $H_{\boldsymbol{p}}$.*

Proof Since the function $\mathbb{S}^{d-1} \to \{0, \dots, n\} : \boldsymbol{p} \mapsto \#\{i : \boldsymbol{p}'\boldsymbol{x}_i \geq 0\}$ takes only finitely many values, a minimizing direction for any dataset $(\boldsymbol{z}, \boldsymbol{X})$ must exist.

First, assume that $\boldsymbol{p}$ is minimizing and $\boldsymbol{p}'\boldsymbol{z} = 0$. By rotating $\boldsymbol{p}$ in the plane defined by $\boldsymbol{p}$ and $\boldsymbol{z}$ by a sufficiently small angle $\alpha > 0$ to the new direction

$$\boldsymbol{p}(\alpha, \boldsymbol{z}) = \cos(\alpha)\boldsymbol{p} + \sin(\alpha)\boldsymbol{z} \tag{23}$$

we can achieve that

$$\boldsymbol{p}(\alpha, \boldsymbol{z})'\boldsymbol{z} = \cos(\alpha)\boldsymbol{p}'\boldsymbol{z} + \sin(\alpha)\boldsymbol{z}'\boldsymbol{z} = \sin(\alpha) > 0,$$

i.e., $\boldsymbol{z} \in H^\circ_{\boldsymbol{p}(\alpha,\boldsymbol{z})}$, while none of the data points $\boldsymbol{x}_i$ that are not in $H_{\boldsymbol{p}}$ enters the halfspace $H_{\boldsymbol{p}(\alpha,\boldsymbol{z})}$. Therefore, since $\boldsymbol{p}$ is minimizing for $(\boldsymbol{z}, \boldsymbol{X})$, so is $\boldsymbol{p}(\alpha, \boldsymbol{z})$.

Now let $\boldsymbol{p}$ be minimizing for $(\boldsymbol{z}, \boldsymbol{X})$ and let $\boldsymbol{p}'\boldsymbol{z} > 0$. Assume that $\boldsymbol{p}'\boldsymbol{x}_{i_0} = 0$ for one of the data points. We rotate $\boldsymbol{p}$ in the plane defined by $\boldsymbol{p}$ and $\boldsymbol{x}_{i_0}$ by a sufficiently small angle α to the new direction $\boldsymbol{p}(\alpha, \boldsymbol{x}_{i_0})$ from (23). Taking $\alpha < 0$ close to 0, we can achieve that

$$\boldsymbol{p}(\alpha, \boldsymbol{x}_{i_0})'\boldsymbol{x}_{i_0} = \cos(\alpha)\boldsymbol{p}'\boldsymbol{x}_{i_0} + \sin(\alpha)\boldsymbol{x}'_{i_0}\boldsymbol{x}_{i_0} = \sin(\alpha) < 0,$$

while $\boldsymbol{z}$ still is in $H^\circ_{\boldsymbol{p}(\alpha,\boldsymbol{x}_{i_0})}$ and none of the data points $\boldsymbol{x}_i$ that were not in $H_{\boldsymbol{p}}$ enters $H_{\boldsymbol{p}(\alpha,\boldsymbol{x}_{i_0})}$. That contradicts the property of $\boldsymbol{p}$ being minimizing, and necessarily no $\boldsymbol{x}_i$ with $\boldsymbol{p}'\boldsymbol{x}_i = 0$ exists.

We have found that a minimizing $\boldsymbol{p}$ such that $\boldsymbol{z} \in H^\circ_{\boldsymbol{p}}$ and none of the data points from $\boldsymbol{X}$ lies on the boundary of $H_{\boldsymbol{p}}$ always exists. □

Lemma 9 *Let $\boldsymbol{p} \in \mathbb{S}^{d-1}$ such that $\boldsymbol{z} \in H^\circ_{\boldsymbol{p}}$ and $0 \leq \dim \operatorname{span}\left(\boldsymbol{X}_{I^0_{\boldsymbol{p}}}\right) = k < d - 1$. Then there exists a data point $\boldsymbol{x}_{i_0} \notin \operatorname{span}\left(\boldsymbol{X}_{I^0_{\boldsymbol{p}}}\right)$ and a vector $\boldsymbol{q} \in \mathbb{S}^{d-1}$ such that $\boldsymbol{z} \in H^\circ_{\boldsymbol{q}}$ and*

$$I^0_{\boldsymbol{q}} = \left[I^0_{\boldsymbol{p}} \cup \{i_0\}\right]^*, \quad I^+_{\boldsymbol{p}} \subset I^+_{\boldsymbol{q}} \cup I^0_{\boldsymbol{q}}, \quad I^-_{\boldsymbol{p}} \subset I^-_{\boldsymbol{q}} \cup I^0_{\boldsymbol{q}}.$$

Further, $\dim \operatorname{span}\left(\boldsymbol{X}_{I^0_{\boldsymbol{q}}}\right) = k + 1$.

In Lemma 9, we are in the situation when the data points from $\boldsymbol{X}$ contained in the boundary of $H_{\boldsymbol{p}}$ do not span the whole set $\partial H_{\boldsymbol{p}}$. Our result says that then, it is possible to change $\boldsymbol{p}$ to $\boldsymbol{q}$ so that at least one additional data point $\boldsymbol{x}_{i_0}$ is on the boundary hyperplane defined by $\boldsymbol{q}$, whereas no other data points from $\boldsymbol{X}$ did change sides (that is, $\operatorname{sign}\left(\boldsymbol{q}'\boldsymbol{x}_i\right) = \operatorname{sign}\left(\boldsymbol{p}'\boldsymbol{x}_i\right)$ or $\operatorname{sign}\left(\boldsymbol{q}'\boldsymbol{x}_i\right) = 0$ for all $i = 1, \dots, n$). At the same time, the halfspace defined by $\boldsymbol{q}$ still contains $\boldsymbol{z}$ in its interior (that is, $\operatorname{sign}\left(\boldsymbol{q}'\boldsymbol{z}\right) = \operatorname{sign}\left(\boldsymbol{p}'\boldsymbol{z}\right) = 1$). In plain words, Lemma 9 guarantees that it is always possible to "tilt" a halfspace $H_{\boldsymbol{p}}$ so that none of the points $\boldsymbol{X} \cup \{\boldsymbol{z}\}$ changes sides of the hyperplane, and some $\boldsymbol{x}_{i_0}$ that was not on the boundary $\partial H_{\boldsymbol{p}}$ of $H_{\boldsymbol{p}}$ moves to $\partial H_{\boldsymbol{q}}$.

Proof The idea of the proof is simple. We start from the direction $\boldsymbol{p}$ and rotate it inside a plane determined by $\boldsymbol{p}$ and $\boldsymbol{r} \in \mathbb{S}^{d-1}$ to get $\boldsymbol{p}(\alpha, \boldsymbol{r})$ from (23). The direction $\boldsymbol{r}$ is taken so that the set of points from $\boldsymbol{X}$ inside the hyperplane $\partial H_{\boldsymbol{p}}$ also remains inside all the rotated hyperplanes $\partial H_{\boldsymbol{p}(\alpha,\boldsymbol{r})}$. We rotate $H_{\boldsymbol{p}}$ this way until its boundary hyperplane hits another point from $\boldsymbol{X}$.

W.l.o.g. assume that $\boldsymbol{p}'\boldsymbol{x}_i = 0$ for $i = 1, \dots, k$ and that $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ are linearly independent. Choose $\boldsymbol{r} \in \operatorname{span}(\boldsymbol{p}, \boldsymbol{x}_1, \dots, \boldsymbol{x}_k)^\perp$. This is possible since $k + 1 < d$. Define $\boldsymbol{p}(\alpha, \boldsymbol{r}) \in \mathbb{S}^{d-1}$ as in (23). Next, we define the set

$$A = \{\alpha \in (-\pi, \pi] : \boldsymbol{p}(\alpha, \boldsymbol{r})'\boldsymbol{z} > 0\}.$$

The set A is composed of all angles α such that $\boldsymbol{z}$ is contained in $H^\circ_{\boldsymbol{p}(\alpha,\boldsymbol{r})}$. Taking $\alpha = 0$ means no rotation is applied, and

$$\boldsymbol{p}(0, \boldsymbol{r})'\boldsymbol{z} = \boldsymbol{p}'\boldsymbol{z} > 0,$$

meaning that $0 \in A$. Certainly, A is thus an open interval of length π that contains 0; denote by $\alpha_0 \in (-\pi, \pi]$ one of the endpoints of this interval. Every data point $\boldsymbol{x}_i$, $i \in I^0_{\boldsymbol{p}}$, is a linear combination of $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ and therefore orthogonal to $\boldsymbol{p}$ as well as $\boldsymbol{r}$. Thus,

$$\boldsymbol{p}(\alpha, \boldsymbol{r})'\boldsymbol{x}_i = 0 \text{ for all } i \in I^0_{\boldsymbol{p}} \text{ and all } \alpha \in (-\pi, \pi]. \tag{24}$$

Now, we rotate $\boldsymbol{p}$ until, for the first time, the boundary hyperplane of $H_{\boldsymbol{p}(\alpha,\boldsymbol{r})}$ meets a data point that is not contained in $\boldsymbol{X}_{I^0_{\boldsymbol{p}}}$. For this, we define

$$\beta = \min\{|\alpha| : \alpha \in A \text{ and } \boldsymbol{p}(\alpha, \boldsymbol{r})'\boldsymbol{x}_i = 0 \text{ for some } i \notin I^0_{\boldsymbol{p}}\} \tag{25}$$

to be the smallest magnitude $|\alpha|$ of an angle $\alpha \in A$ such that there is a new data point from $\boldsymbol{X}$ (that is, a point from $\boldsymbol{X}$ not contained already in $\partial H_{\boldsymbol{p}}$) inside the boundary of $H_{\boldsymbol{p}(\alpha,\boldsymbol{r})}$. We first need to show that such an angle α exists, that is β is well defined.

Take first the boundary point α_0 of the interval A. There must be at least one data point $\boldsymbol{x}_{i_0}$ with $i_0 \notin I^0_{\boldsymbol{p}}$ such that

$$\boldsymbol{p}(\alpha_0, \boldsymbol{r})'\boldsymbol{x}_{i_0} \neq 0. \tag{26}$$

This is true because if no such index $i_0 \notin I^0_{\boldsymbol{p}}$ existed, we would have, thanks to (24), that $\boldsymbol{p}(\alpha_0, \boldsymbol{r})'\boldsymbol{x}_i = 0$ for all $i = 1, \dots, n$, and necessarily all data points from $\boldsymbol{X}$ would lie

in the hyperplane $\partial H_{\boldsymbol{p}(\alpha_0,\boldsymbol{r})}$. That, however, contradicts our assumption span $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)=\mathbb{R}^d$.

There exist two angles $\alpha_1,\alpha_2\in(-\pi,\pi]$ with $\alpha_1+\pi=\alpha_2$ such that $\boldsymbol{p}(\alpha_j,\boldsymbol{r})'\boldsymbol{x}_{i_0}=0$ for both $j=1,2$. Since $\boldsymbol{p}'\boldsymbol{x}_{i_0}\neq 0$, this follows from the π-periodicity of the cotangent and

$$\begin{aligned}\boldsymbol{p}(\alpha,\boldsymbol{r})'\boldsymbol{x}_{i_0}=0 &\iff \cos(\alpha)\boldsymbol{p}'\boldsymbol{x}_{i_0}+\sin(\alpha)\boldsymbol{r}'\boldsymbol{x}_{i_0}=0\\ &\iff \cot(\alpha)=-\frac{\boldsymbol{r}'\boldsymbol{x}_{i_0}}{\boldsymbol{p}'\boldsymbol{x}_{i_0}}\in\mathbb{R}.\end{aligned}$$

Since the interval A has length π and $\alpha_0\neq\alpha_1$, $\alpha_0\neq\alpha_2$ because of (26), exactly one of the angles α_1,α_2 must lie inside the interval A. Therefore, the set from which β is taken as the minimum in (25) is non-empty and finite, and β itself is well-defined, as we wanted to show.

It remains to choose $\boldsymbol{q}=\boldsymbol{p}(\beta,\boldsymbol{r})$ or $\boldsymbol{q}=\boldsymbol{p}(-\beta,\boldsymbol{r})$, depending on whether the minimal angle $\alpha\neq 0$ such that $|\alpha|=\beta$ in (25) is positive, or negative. We obtain that $\boldsymbol{q}'\boldsymbol{x}_{i_0}=0$ for some $i_0\notin I_{\boldsymbol{p}}^0$, while still $\boldsymbol{q}'\boldsymbol{z}>0$. Then $\boldsymbol{q}$ and $\boldsymbol{x}_{i_0}$ satisfy the desired properties. □

Lemma 10 *If $\boldsymbol{p}\in\mathbb{S}^{d-1}$ is minimizing for $(\boldsymbol{z},\boldsymbol{X})$ with $I_{\boldsymbol{p}}^0=\emptyset$ and $\boldsymbol{z}\in H_{\boldsymbol{p}}^\circ$, then for every k, $1\le k<d$, there are k linearly independent data points $\boldsymbol{x}_{i_1},\ldots,\boldsymbol{x}_{i_k}$ and a vector $\boldsymbol{q}\in\mathbb{S}^{d-1}$ such that $\boldsymbol{z}\in H_{\boldsymbol{q}}^\circ$ and*

$$I_{\boldsymbol{q}}^0=\{i_1,\ldots,i_k\}^*,\quad I_{\boldsymbol{p}}^+\subset I_{\boldsymbol{q}}^+\cup I_{\boldsymbol{q}}^0,\quad I_{\boldsymbol{p}}^-\subset I_{\boldsymbol{q}}^-\cup I_{\boldsymbol{q}}^0.$$

Proof The proposition follows from repeated application of Lemma 9. □

We have gathered all the prerequisites and are ready to start with the main proof of Theorem 5. First, in Lemmas 11 and 12 we prove two inequalities. They together give that for each k such that $1\le k<d$ we have

$$\begin{aligned}&ahD(\boldsymbol{z}|\boldsymbol{X})\\ &=\min_{I\in\mathcal{L}_k}\left[\left(\min_{\boldsymbol{p}\in\mathcal{S}(\boldsymbol{z},\boldsymbol{X}_I)}\left(n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0\right)\right)-\left(n_{I^*}-hD(\boldsymbol{0}|\boldsymbol{X}_{I^*})\right)\right],\end{aligned}\tag{27}$$

where $\mathcal{S}(\boldsymbol{z},\boldsymbol{X}_I)=\{\boldsymbol{p}\in\mathbb{S}^{d-1}\colon \boldsymbol{p}\in\operatorname{span}(\boldsymbol{X}_I)^\perp,\boldsymbol{z}\in H_{\boldsymbol{p}}^\circ\}$ is the collection of unit normals $\boldsymbol{p}$ such that $\boldsymbol{X}_I\subset\partial H_{\boldsymbol{p}}$ and $\boldsymbol{z}\in H_{\boldsymbol{p}}^\circ$. Then, in Lemma 13 it is shown that the inner minimum on the right-hand side of (27) can be written as

$$\min_{\boldsymbol{p}\in\mathcal{S}(\boldsymbol{z},\boldsymbol{X}_I)}\left(n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0\right)=ahD(\boldsymbol{Q}_I'\boldsymbol{z}|\boldsymbol{Q}_I'\boldsymbol{X}_{(I^*)^c})+n_{I^*},$$

and the last term in (27) can be expressed as

$$hD(\boldsymbol{0}|\boldsymbol{X}_{I^*})=hD(\boldsymbol{0}|\boldsymbol{P}_I'\boldsymbol{X}_{I^*}).$$

Putting all together, we obtain the desired claim of Theorem 5.

Lemma 11 *For any k, $1\le k<d$, let $\boldsymbol{p}\in\mathbb{S}^{d-1}$ be such that $\boldsymbol{z}\in H_{\boldsymbol{p}}^\circ$ and $I=\{i_1,\ldots,i_k\}\subset I_{\boldsymbol{p}}^0$, that is let $\boldsymbol{p}\in\mathcal{S}(\boldsymbol{z},\boldsymbol{X}_I)$. Then,*

$$ahD(\boldsymbol{z}|\boldsymbol{X})\le(n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0)-\left(n_{I^*}-hD(\boldsymbol{0}|\boldsymbol{X}_{I^*})\right).\tag{28}$$

Proof Before starting with the proof, we introduce the necessary terminology. For a dataset $\boldsymbol{X}$ of points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$ in $\mathbb{R}^d$ we say that a direction $\boldsymbol{p}\in\mathbb{S}^{d-1}$ is hD-minimizing for the dataset $(\boldsymbol{0},\boldsymbol{X})$ if

$$hD(\boldsymbol{0}|\boldsymbol{X})=\#\left\{i\colon \boldsymbol{p}'\boldsymbol{x}_i\ge 0\right\}=\#\left\{i\colon \boldsymbol{x}_i\in H_{\boldsymbol{p}}\right\},$$

for hD the (Euclidean) halfspace depth defined in (1). Just as for the angular halfspace depth ahD in Lemma 8, also for the standard halfspace depth hD it is straightforward to see that an hD-minimizing direction $\boldsymbol{p}$ with no data points in the hyperplane $\partial H_{\boldsymbol{p}}$—except for possible data points located exactly at $\boldsymbol{0}\in\mathbb{R}^d$—always exists (Dyckerhoff and Mozharovskyi 2016, Proposition 1, Laketa and Nagy 2021, Lemma 3).

To prove (28), we use a tilting argument again. First, we explain its idea and then proceed with the formal proof. The halfspace $H_{\boldsymbol{p}}$ is slightly perturbed to $H_{\boldsymbol{q}}$ in a way that all points not in the boundary of $H_{\boldsymbol{p}}$ remain on the same side of the hyperplanes $\partial H_{\boldsymbol{p}}$ and $\partial H_{\boldsymbol{q}}$. Inside the hyperplane $\partial H_{\boldsymbol{p}}$, we first consider the (Euclidean) halfspace depth $hD(\boldsymbol{0}|\boldsymbol{X}_{I^*})$ and find a minimizing direction $\boldsymbol{r}\in\mathbb{S}^{d-1}$ for $(\boldsymbol{0},\boldsymbol{X}_{I^*})$. We tilt $\boldsymbol{p}$ in such a way that inside the linear hull span $(\boldsymbol{X}_I)$ of $\boldsymbol{X}_I$, only those data points lying inside the hD-minimizing halfspace $H_{\boldsymbol{r}}$ lie inside $H_{\boldsymbol{q}}$. This way, after the tilt, the halfspace $H_{\boldsymbol{q}}$ contains the $n_{\boldsymbol{p}}^+$ data points inside $H_{\boldsymbol{p}}^\circ$, and all $n_{\boldsymbol{p}}^0$ data points from $\partial H_{\boldsymbol{p}}$, except for those that were "tilted out" from the boundary $\partial H_{\boldsymbol{p}}$. We have constructed $H_{\boldsymbol{q}}$ in such a way that exactly $n_{I^*}-hD(\boldsymbol{0}|\boldsymbol{X}_{I^*})$ data points are "tilted out" by passing from $H_{\boldsymbol{p}}$ to $H_{\boldsymbol{q}}$, giving the final bound (28).

Having explained the idea of the proof, we write it down formally. Let the direction $\boldsymbol{r}\in\operatorname{span}\left(\boldsymbol{x}_{i_1},\ldots,\boldsymbol{x}_{i_k}\right)\cap\mathbb{S}^{d-1}$ be hD-minimizing for the dataset $(\boldsymbol{0},\boldsymbol{X}_{I^*})$. Consider the tilted direction $\boldsymbol{q}=\boldsymbol{p}(\alpha,\boldsymbol{r})$ as in (23) with $\alpha>0$ small. We have

$$\boldsymbol{q}'\boldsymbol{x}_i=\cos(\alpha)\boldsymbol{p}'\boldsymbol{x}_i+\sin(\alpha)\boldsymbol{r}'\boldsymbol{x}_i.$$

By choosing α sufficiently small, we can guarantee that $\boldsymbol{q}'\boldsymbol{x}_i$ and $\boldsymbol{p}'\boldsymbol{x}_i$ share the same sign for all $i\in I_{\boldsymbol{p}}^+\cup I_{\boldsymbol{p}}^-$. Similarly, since we assumed that $\boldsymbol{z}\in H_{\boldsymbol{p}}^\circ$, we can also suppose that the signs of $\boldsymbol{q}'\boldsymbol{z}$ and $\boldsymbol{p}'\boldsymbol{z}$ are the same. This guarantees that all data points and $\boldsymbol{z}$ lie on the same side of $\partial H_{\boldsymbol{p}}$ and $\partial H_{\boldsymbol{q}}$. Because $I\subset I_{\boldsymbol{p}}^0$, for $\boldsymbol{x}_i\in\boldsymbol{X}_{I^*}$ it holds that $\boldsymbol{q}'\boldsymbol{x}_i=\sin(\alpha)\boldsymbol{r}'\boldsymbol{x}_i$. Since $\boldsymbol{r}$ is hD-minimizing for $(\boldsymbol{0},\boldsymbol{X}_{I^*})$, there are exactly $n_{I^*}-hD(\boldsymbol{0}|\boldsymbol{X}_{I^*})$ data points in $\boldsymbol{X}_{I^*}$ for which $\boldsymbol{r}'\boldsymbol{x}_i<0$.

Therefore,

$$\begin{aligned} ahD(z|X) &\le \#\left\{i : x_i \in H_q\right\} = \#\{i : q'x_i \ge 0\} \\ &= \#\{i : p'x_i \ge 0\} - \#\{i \in I^* : r'x_i < 0\} \\ &= (n_p^+ + n_p^0) - \left(n_{I^*} - hD(\mathbf{0}|X_{I^*})\right), \end{aligned}$$

as we wanted to prove. □

Lemma 12 *For every k, $1 \le k < d$, there is a set $I = \{i_1, \ldots, i_k\}$ of indices and a vector $p \in \mathbb{S}^{d-1}$ such that $x_{i_1}, \ldots, x_{i_k}$ are linearly independent, $I_p^0 = I^*$, $z \in H_p^\circ$, and*

$$ahD(z|X) \ge (n_p^+ + n_p^0) - \left(n_{I^*} - hD(\mathbf{0}|X_{I^*})\right). \tag{29}$$

In other words, for every k, $1 \le k < d$ there exists $I \in \mathcal{L}_k$ and $p \in \mathcal{S}(z, X_I)$ such that (29) is true.

Proof Let $q \in \mathbb{S}^{d-1}$ be minimizing for (z, X) such that $I_q^0 = \emptyset$ and $z \in H_q^\circ$. Such a direction q exists according to Lemma 8. Choose p as in Lemma 10, that is, there are k linearly independent points from X on the boundary of H_p, and $\operatorname{sign}\left(p'x_i\right) = \operatorname{sign}\left(q'x_i\right)$ or $\operatorname{sign}\left(p'x_i\right) = 0$ for all $i = 1, \ldots, n$. For $i \in I_q^- \cap I_p^0$ it holds that $q'x_i < 0$, which implies

$$hD(\mathbf{0}|X_{I_p^0}) \le n_p^0 - \#(I_q^- \cap I_p^0).$$

Then,

$$\begin{aligned} ahD(z|X) &= \#\{i : q'x_i > 0\} = \#\{i : p'x_i \ge 0\} - \#(I_q^- \cap I_p^0) \\ &\ge (n_p^+ + n_p^0) - \left(n_p^0 - hD(\mathbf{0}|X_{I_p^0})\right). \end{aligned}$$

Because of Lemma 10, it holds that $I_p^0 = I^*$ and thus $n_p^0 = n_{I^*}$, which completes the proof. □

Lemma 13 *For every k, $1 \le k < d$, and $I \in \mathcal{L}_k$ we have*

$$\min_{p \in \mathcal{S}(z, X_I)} \left(n_p^+ + n_p^0\right) = ahD(Q_I'z | Q_I'X_{(I^*)^c}) + n_{I^*}, \tag{30}$$

and

$$hD(\mathbf{0}|X_{I^*}) = hD(\mathbf{0}|P_I'X_{I^*}). \tag{31}$$

Proof Let $I = \{i_1, \ldots, i_k\} \in \mathcal{L}_k$ and recall that Q_I is the matrix whose $d - k$ columns are vectors of an orthonormal basis of the orthogonal complement of span $\left(x_{i_1}, \ldots, x_{i_k}\right)$. Thus, every vector $p \in \operatorname{span}(X_I)^\perp \cap \mathbb{S}^{d-1}$ is a linear combination of the columns of Q_I, i.e., $p = Q_I q$ for some $q \in \mathbb{S}^{d-k-1}$, and the map $\mathbb{S}^{d-k-1} \to \mathbb{S}^{d-1} \cap \operatorname{span}(X_I)^\perp$, $q \mapsto Q_I q$, is a bijection. Moreover, $z \in H_p^\circ$ if and only if $Q_I'z \in H_q^\circ$. Since

$$q'(Q_I'x_i) \ge 0 \quad \Longleftrightarrow \quad (q'Q_I')x_i \ge 0 \quad \Longleftrightarrow \quad p'x_i \ge 0$$

for $p = Q_I q$, we conclude

$$\begin{aligned} &\min_{p \in \mathcal{S}(z, X_I)} (n_p^+ + n_p^0) \\ &= \min_{p \in \mathcal{S}(z, X_I)} \#\{i : p'x_i \ge 0\} \\ &= \min_{q \in \mathbb{S}^{d-k-1},\, Q_I'z \in H_q^\circ} \#\{i : q'(Q_I'x_i) \ge 0\}. \end{aligned}$$

All points $x_i, i \in I^*$, are mapped to the origin by Q_I'. Hence, $q'(Q_I'x_i) = 0$ for $i \in I^*$ and it follows that

$$\begin{aligned} &\min_{p \in \mathcal{S}(z, X_I)} (n_p^+ + n_p^0) \\ &= \min_{q \in \mathbb{S}^{d-k-1},\, Q_I'z \in H_q^\circ} \#\{i \notin I^* : q'(Q_I'x_i) \ge 0\} + n_I^*. \end{aligned}$$

Since $Q_I'x_i \ne \mathbf{0}$ for $i \notin I^*$, the minimum on the right-hand side is simply the angular halfspace depth of $Q_I'z$ w.r.t. the dataset $Q_I'x_i$, $i \notin I^*$. As noted in Remark 4, the fact that neither the $Q_I'z$ nor the $Q_I'x_i$, $i \notin I^*$, are in general unit vectors is not a problem. Therefore, we have proved (30). Formula (31) is proved completely analogously. □

Appendix C Additional tables with numerical results

Table 11 Computation times for computing the angular halfspace depth of a single point z w.r.t. a sample of size n from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$ using algorithm `C2(M)`. Reported are the times for the algorithm running on a single CPU core and on 4 CPU cores as well as the respective ratio. All points are in general position. Times are averaged over 10 independent runs. They are given in seconds with three significant digits. Ratios between 3 and 35 are marked in yellow, ratios of at least 35 in orange

d	1 point	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	1 Core	0.0003	0.00075	0.00292	0.011	0.0436	0.18	0.786	3.18	12.7	61.3	239	1010
	4 Cores	0.00018	0.00038	0.00102	0.00342	0.0172	0.0563	0.209	1.09	3.72	14.6	60.6	278
	ratio	1.67	1.97	2.86	3.22	2.53	3.19	3.75	2.92	3.41	4.2	3.94	3.62
4	1 Core	0.0039	0.0277	0.212	1.78	13.9	115	950					
	4 Cores	0.0014	0.00887	0.0602	0.469	4.03	31.2	257					
	ratio	2.79	3.12	3.52	3.79	3.47	3.68	3.7					
5	1 Core	0.0534	0.766	12	188	3060							
	4 Cores	0.0161	0.237	3.68	51.6	821							
	ratio	3.32	3.23	3.25	3.64	3.73							
6	1 Core	0.586	16.3	497									
	4 Cores	0.171	5.37	139									
	ratio	3.42	3.03	3.57									
7	1 Core	4.49	260										
	4 Cores	1.3	77.8										
	ratio	3.45	3.35										
8	1 Core	27.5	3730										
	4 Cores	7.76	1020										
	ratio	3.55	3.65										
9	1 Core	148											
	4 Cores	40.3											
	ratio	3.68											
10	1 Core	670											
	4 Cores	182											
	ratio	3.69											

Table 12 Computation times for computing the angular halfspace depth of $m = 1\,000$ points $z_1, \ldots, z_{1000}$ w.r.t. a sample of size n from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$ using algorithm `C2(M)`. Reported are the times for the algorithm running on a single CPU core and on 4 CPU cores as well as the respective ratio. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits. Ratios between 3 and 3.5 are marked in yellow, ratios of at least 3.5 in orange

d	1000 points	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	1 Core	0.00308	0.00697	0.0165	0.0421	0.109	0.315	1.11	3.83	14.1	55.7	244	1020
	4 Cores	0.00095	0.00204	0.00474	0.0111	0.0295	0.0881	0.282	1.19	4.12	15.5	63.3	288
	ratio	3.24	3.42	3.48	3.79	3.69	3.58	3.92	3.22	3.43	3.59	3.85	3.56
4	1 Core	0.0598	0.274	1.41	6.66	34.5	202	1330					
	4 Cores	0.0167	0.0749	0.353	2.27	10.3	54.5	358					
	ratio	3.59	3.65	4	2.93	3.36	3.72	3.72					
5	1 Core	0.794	7.58	71.9	711								
	4 Cores	0.227	2.31	19.5	190								
	ratio	3.5	3.29	3.69	3.74								
6	1 Core	8.04	153	3000									
	4 Cores	2.41	41.2	805									
	ratio	3.33	3.72	3.73									
7	1 Core	58.4	2390										
	4 Cores	15.9	649										
	ratio	3.67	3.68										
8	1 Core	343											
	4 Cores	92.2											
	ratio	3.72											
9	1 Core	1750											
	4 Cores	468											
	ratio	3.74											

Table 13 Computation times for computing the angular halfspace depth of all n data points in a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$ w.r.t. the sample itself using algorithm `C2(M)`. Reported are the times for the algorithm running on a single CPU core and on 4 CPU cores as well as the respective ratio. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits. Ratios between 3 and 3.5 are marked in yellow, ratios of at least 3.5 in orange

d	sample	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	1 Core	0.00024	0.00089	0.00318	0.012	0.0477	0.197	0.797	3.47	14	67.1	278	1170
	4 Cores	0.00012	0.00035	0.0011	0.0041	0.0138	0.0572	0.224	0.911	4.11	16.4	70.3	341
	ratio	2	2.54	2.89	2.93	3.46	3.45	3.56	3.81	3.4	4.1	3.95	3.44
4	1 Core	0.00421	0.0316	0.236	1.94	15.2	125	1030					
	4 Cores	0.00135	0.00907	0.0649	0.505	4.42	33.8	279					
	ratio	3.12	3.49	3.63	3.86	3.45	3.7	3.71					
5	1 Core	0.0573	0.824	12.9	207	3350							
	4 Cores	0.0159	0.224	3.76	55.8	897							
	ratio	3.61	3.67	3.43	3.7	3.73							
6	1 Core	0.617	17.4	534									
	4 Cores	0.162	4.98	144									
	ratio	3.81	3.5	3.72									
7	1 Core	4.74	278										
	4 Cores	1.41	75.4										
	ratio	3.36	3.68										
8	1 Core	29.1	3650										
	4 Cores	8.14	985										
	ratio	3.57	3.71										
9	1 Core	155											
	4 Cores	42.4											
	ratio	3.67											
10	1 Core	702											
	4 Cores	189											
	ratio	3.72											

Table 14 Computation times for computing the angular halfspace depth of a single point z (Task 1), for $m = 1\,000$ points $z_1, \ldots, z_m$ (Task 2) or all data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in the sample (Task 3) w.r.t. a sample of size n from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$ using algorithm `C2(M)` on a modern CPU (Intel® Core™ i5-13600) with 6 P-cores and 8 E-cores using all 14 cores. All points are in general position. Times are averaged over 10 independent runs and are given in seconds with three significant digits. The results in this table should be compared with Tables 11–13

d		40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
3	Task 1	0.00008	0.00023	0.00049	0.00153	0.0051	0.0194	0.0747	0.286	1.14	4.61	19.1	85.6
	Task 2	0.0004	0.0009	0.00199	0.0045	0.0112	0.0337	0.104	0.352	1.26	4.82	19.8	88.5
	Task 3	0.00007	0.00024	0.00064	0.0019	0.0062	0.0238	0.0845	0.3264	1.313	5.367	22.25	103.4
4	Task 1	0.00044	0.00325	0.02	0.155	1.18	9.36	77.3					
	Task 2	0.00647	0.0286	0.129	0.572	2.99	17.3	115					
	Task 3	0.00048	0.0037	0.0252	0.174	1.37	10.8	89.7					
5	Task 1	0.00486	0.0772	0.97	15	244							
	Task 2	0.0767	0.686	6.38	59.8								
	Task 3	0.00572	0.0783	1.11	17.9	284							
6	Task 1	0.0482	1.47	37.6									
	Task 2	0.68	13	241									
	Task 3	0.0534	1.53	43.2									
7	Task 1	0.37	24.6										
	Task 2	4.96	195										
	Task 3	0.386	23.6										
8	Task 1	2.07	292										
	Task 2	28.3											
	Task 3	2.39	305										
9	Task 1	10.8											
	Task 2	139											
	Task 3	12.1											
10	Task 1	54											
	Task 2												
	Task 3	52.7											

References

Agostinelli, C., Romanazzi, M.: Nonparametric analysis of directional data based on data depth. Environ. Ecol. Stat. **20**(2), 253–270 (2013)

Buttarazzi, D., Pandolfo, G., Porzio, G.C.: A boxplot for circular data. Biometrics **74**(4), 1492–1501 (2018)

Cascos, I.: The expected convex hull trimmed regions of a sample. Comput. Statist. **22**(4), 557–569 (2007)

Cascos, I., Ochoa, M.: Expectile depth: theory and computation for bivariate datasets. J. Multivariate Anal. **184**, 104757 (2021)

Chen, D., Morin, P., Wagner, U.: Absolute approximation of Tukey depth: theory and experiments. Comput. Geom. **46**(5), 566–573 (2013)

Chernozhukov, V., Galichon, A., Hallin, M., Henry, M.: Monge-Kantorovich depth, quantiles, ranks and signs. Ann. Statist. **45**(1), 223–256 (2017)

Donoho, D.L., Gasko, M.: Breakdown properties of location estimates based on halfspace depth and projected outlyingness. Ann. Statist. **20**(4), 1803–1827 (1992)

Dudley, R. M.: Real Analysis and Probability, volume 74 of Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge. Revised reprint of the 1989 original, (2002)

Dyckerhoff, R.: Computing zonoid trimmed regions of bivariate data sets. In: Bethlehem, J., Heijden, P. (eds.) Compstat 2000 - Proceedings in Computational Statistics, pp. 295–300. Physica, Heidelberg (2000)

Dyckerhoff, R.: Data depths satisfying the projection property. Allg. Stat. Arch. **88**(2), 163–190 (2004)

Dyckerhoff, R., Mozharovskyi, P.: Exact computation of the halfspace depth. Comput. Statist. Data Anal. **98**, 19–30 (2016)

Dyckerhoff, R., Mozharovskyi, P., Nagy, S.: Approximate computation of projection depths. Comput. Statist. Data Anal. **157**, 107166 (2021)

Edelsbrunner, H.: Algorithms in Combinatorial Geometry, volume 10 of EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, (1987)

Genest, M., Massé, J.-C., Plante, J.-F.: depth: Nonparametric depth functions for multivariate analysis. R package version 2(1–1), 1 (2019)

Golub, G. H., Van Loan, C. F.: Matrix Computations. The Johns Hopkins University Press, Baltimore, MD, fourth edition, (2013)

Hallin, M., Liu, H., Verdebout, T.: Nonparametric measure-transportation-based methods for directional data. J. R. Stat. Soc. Ser. B Stat Methodol. **86**(5), 1172–1196 (2024)

Konen, D., Paindaveine, D.: Spatial quantiles on the hypersphere. Ann. Statist. **51**(5), 2221–2245 (2023)

Koshevoy, G.A., Mosler, K.: Zonoid trimming for multivariate distributions. Ann. Statist. **25**(5), 1998–2017 (1997)

Kreher, D.L., Stinson, D.R.: Combinatorial Algorithms: Generation, Enumeration, and Search. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton (1999)

Laketa, P., Nagy, S.: Reconstruction of atomic measures from their halfspace depth. J. Multivariate Anal. **183**, 104727 (2021)

Ley, C., Sabbah, C., Verdebout, T.: A new concept of quantiles for directional data and the angular Mahalanobis depth. Electron. J. Stat. **8**(1), 795–816 (2014)

Ley, C., Verdebout, T.: Modern Directional Statistics. Chapman & Hall/CRC Interdisciplinary Statistics Series. CRC Press, Boca Raton, FL (2017)

Li, J., Cuesta-Albertos, J.A., Liu, R.Y.: DD-classifier: nonparametric classification procedure based on DD-plot. J. Amer. Statist. Assoc. **107**(498), 737–753 (2012)

Li, J., Liu, R.Y.: New nonparametric tests of multivariate locations and scales using data depth. Statist. Sci. **19**(4), 686–696 (2004)

Liu, R.Y.: On a notion of data depth based on random simplices. Ann. Statist. **18**(1), 405–414 (1990)

Liu, R.Y., Parelius, J.M., Singh, K.: Multivariate analysis by data depth: descriptive statistics, graphics and inference. Ann. Statist. **27**(3), 783–858 (1999)

Liu, R.Y., Singh, K.: Ordering directional data: concepts of data depth on circles and spheres. Ann. Statist. **20**(3), 1468–1484 (1992)

Liu, X., Mosler, K., Mozharovskyi, P.: Fast computation of Tukey trimmed regions and median in dimension $p>2$. J. Comput. Graph. Statist. **28**(3), 682–697 (2019)

Mardia, K. V.: Statistics of Directional Data. Academic Press, London-New York. Probability and Mathematical Statistics, No. 13, (1972)

Mardia, K.V., Jupp, P.E.: Directional Statistics. Wiley Series in Probability and Statistics. John Wiley & Sons, Ltd., Chichester (2000)

Miller, K., Ramaswami, S., Rousseeuw, P., Sellarès, J.A., Souvaine, D., Streinu, I., Struyf, A.: Efficient computation of location depth contours by methods of computational geometry. Stat. Comput. **13**(2), 153–162 (2003)

Mosler, K., Mozharovskyi, P.: Choosing among notions of multivariate depth statistics. Statist. Sci. **37**(3), 348–368 (2022)

Nagy, S., Demni, H., Buttarazzi, D., Porzio, G.C.: Theory of angular depth for classification of directional data. Adv. Data Anal. Classif. **18**, 627–662 (2024)

Nagy, S., Dyckerhoff, R., Mozharovskyi, P.: Uniform convergence rates for the approximated halfspace and projection depth. Electron. J. Stat. **14**(2), 3939–3975 (2020)

Nagy, S., Laketa, P.: Theoretical properties of angular halfspace depth. Bernoulli **31**(2), 1007–1031 (2025)

Pandolfo, G., D'Ambrosio, A., Porzio, G.C.: A note on depth-based classification of circular data. Electron. J. Appl. Stat. Anal. **11**(2), 447–462 (2018)

Pandolfo, G., Paindaveine, D., Porzio, G.C.: Distance-based depths for directional data. Canad. J. Statist. **46**(4), 593–609 (2018)

Pokotylo, O., Mozharovskyi, P., Dyckerhoff, R.: Depth and depth-based classification with R package ddalpha. J. Stat. Softw. **91**(5), 1–46 (2019)

Pokotylo, O., Mozharovskyi, P., Dyckerhoff, R., Nagy, S.: ddalpha: Depth-based classification and calculation of data depth. R package version 1(3), 16 (2024)

R Core Team (2024). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria

Rousseeuw, P.J., Ruts, I., Tukey, J.W.: The bagplot: A bivariate boxplot. Am. Stat. **53**(4), 382–387 (1999)

Rousseeuw, P.J., Struyf, A.: Computing location depth and regression depth in higher dimensions. Stat. Comput. **8**(3), 193–203 (1998)

Ruts, I., Rousseeuw, P.J.: Computing depth contours of bivariate point clouds. Comput. Statist. Data Anal. **23**(1), 153–168 (1996)

Small, C.G.: Measures of centrality for multivariate and directional distributions. Canad. J. Statist. **15**(1), 31–39 (1987)

Tukey, J. W.: Mathematics and the picturing of data. In Proceedings of the International Congress of Mathematicians (Vancouver, B. C., 1974), Vol. 2, pages 523–531. Canad. Math. Congress, Montreal, Que, (1975)

Watson, G. S.: Statistics on Spheres, volume 6 of University of Arkansas Lecture Notes in the Mathematical Sciences. John Wiley & Sons, Inc., New York. A Wiley-Interscience Publication, (1983)

Zuo, Y., Serfling, R.: General notions of statistical depth function. Ann. Statist. **28**(2), 461–482 (2000)