Stat Comput (2014) 24:51–63
DOI 10.1007/s11222-012-9352-6

Computing projection depth and its associated estimators

Xiaohui Liu · Yijun Zuo

Received: 27 December 2011 / Accepted: 29 August 2012 / Published online: 29 September 2012

Abstract To facilitate the application of projection depth, an exact algorithm is proposed from the view of cutting a convex polytope with hyperplanes. Based on this algorithm, one can obtain a finite number of optimal direction vectors, which are x-free and therefore enable us (Liu et al., Preprint, 2011) to compute the projection depth and most of its associated estimators of dimension $p \geq 2$, including Stahel-Donoho location and scatter estimators, projection trimmed mean, projection depth contours and median, etc. Both real and simulated examples are also provided to illustrate the performance of the proposed algorithm.

Keywords Exact algorithm · Projection depth · Stahel-Donoho estimators · Projection trimmed mean · Projection depth contours · Projection median

1 Introduction

In multivariate analysis, depth functions are very powerful because they can offer very effective ways to extract relevant information from data. Based on the depth functions, a lot of methods of signs and ranks, order statistics, quantiles, and outlyingness measures could be extended conveniently from their univariate counterparts in a unified way (Serfling 2006). Owing to these favorable properties, several notions of depth have been proposed in the last decades since the seminal work of Tukey (1975). To name a few, halfspace depth (Tukey 1975), simplicial depth (Liu 1990), regression depth (Rousseeuw and Hubert 1999) and projection depth (Liu 1992; Zuo and Serfling 2000; Zuo 2003).

Among these notions, projection depth (PD) appears to be very favorable. For example, it satisfies all the desirable properties of the general statistical depth function defined in Zuo and Serfling (2000), namely, affine invariance, maximality at center, monotonicity relative to deepest point, and vanishing at infinity. Furthermore, with a proper choice of (μ, σ), it can induce a lot of favorable estimators, such as Stahel-Donoho location and scatter estimators, which enjoy very high breakdown point robustness and affine equivariance (Zuo et al. 2004; Zuo 2006), and projection median, which has the highest breakdown point among all the existing affine equivariant multivariate location estimators (Zuo 2003). In addition, the projection depth regions are continuous and not vanishing even outside the convex hull of the data cloud, therefore it can be extended to develop some desirable classifiers (Hubert and Van der Veeken 2010). Moreover, it turns out that (Liu et al. 2011), under mild conditions, the depth value of any given data point x with respect to (w.r.t.) a fixed data cloud depends only on a finite number of direction vectors, which do not depend on the point x for which the depth value is being computed, and are determined completely by the fixed data cloud, i.e. x-free. Without loss of generality, they are called optimal direction vectors. Based on these direction vectors, one can conveniently compute the projection depth and its induced estimators, especially, the projection depth contours and projection median by using the common linear programming.

However, it is not trivial to obtain all the optimal direction vectors. The computation becomes even more challenging due to the existence of the absolute value symbols in

X. Liu
School of Statistics, Jiangxi University of Finance and Economics, Nanchang, Jiangxi 330013, China

X. Liu · Y. Zuo (✉)
Department of Statistics and Probability, Michigan State University, East Lansing, MI 48823, USA
e-mail: zuo@msu.edu

MAD. As a result, an exact algorithm exists only for bivariate data (Zuo and Lai 2011). To facilitate the application of projection depth, a new exact algorithm is proposed. It is noteworthy that, since the new algorithm modifies the way of handling the absolute value symbols developed in Zuo and Lai (2011), and employs the method of cutting a convex polytope, i.e. cone, with hyperplanes, it is feasible to exactly compute the projection depth and its associated estimators in more general spaces with dimension higher than 2. It is found that the proposed algorithm is efficient, and much faster than that of Zuo and Lai (2011) when $p=2$. The idea of how to find all the cones is motivated mainly by Paindaveine and Šiman (2012a), which developed a very efficient way to compute the halfspace depth contour through linear programming for $p \geq 2$; see also Paindaveine and Šiman (2011, 2012b). The implementation (Matlab codes) of our new algorithm can be obtained from the corresponding author (zuo@msu.edu).

The rest of the paper is organized as follows. Section 2 describes the methodology of the projection depth and its associated estimators. Section 3 provides the main idea of the exact algorithm. Section 4 proposes the exact algorithm. Section 5 examines the performance of the proposed algorithm through some examples, and provides some empirical results about its computational time complexity.

2 Projection depth and its associated estimators

For a given distribution F_1 on R^1, let $\mu(F_1)$ be translation and scale equivariant, and $\sigma(F_1)$ be translation invariant and scale equivariant. Then the projection depth value of a given point $x \in R^p$ w.r.t. F can be defined as (Liu 1992; Zuo and Serfling 2000)

$$PD(x, F) = \frac{1}{1 + O(x, F)},$$

where

$$O(x, F) = \sup_{u \in \mathcal{S}^{p-1}} \left| Q(u, x, F) \right|, \tag{1}$$

where $\mathcal{S}^{p-1} = \{v \in R^p : \|v\| = 1\}$ with $\|\cdot\|$ standing for the Euclidean norm, and $Q(u, x, F) = (u^T x - \mu(F_u))/\sigma(F_u)$. F_u is the distribution of $u^T X$, which denotes the projection of X onto the unit vector u. Without loss of generality, we define $Q(u, x, F) = 0$ if $u^T x - \mu(F_u) = \sigma(F_u) = 0$ throughout this paper.

Note that the projection depth and its associated estimators depend on the choice of $(\mu(F), \sigma(F))$. Different choices of $\mu(F)$ and $\sigma(F)$ can result in different estimators in terms of robustness and efficiency. Therefore, in what follows, we select $(\mu(F), \sigma(F))$ as the commonly used robust choice (Med, MAD), where Med and MAD denote the median and median absolute deviation, respectively.

Similar to some other notions of depth, one major function of the projection depth is to provide a center-outward ordering for the multivariate data. Based on this ordering, one can induce the projection depth contour, which can provide us with a multivariate view of the quantile of an underlying distribution (Halin et al. 2010). Write

$$PR(\alpha, F) = \left\{ x \in R^p : PD(x, F) \geq \alpha \right\}$$

as the α-th projection depth region, where $0 \leq \alpha \leq \alpha^* = \sup_{x \in R^p} PD(x, F)$. Then the corresponding α-th projection depth contour can be defined as the boundary of $PR(\alpha, F)$, that is,

$$PC(\alpha, F) = \left\{ x \in R^p : PD(x, F) = \alpha \right\}.$$

As proved in Zuo (2003), these depth regions can form a sequence of nested convex sets: each $PR(\alpha, F)$ is convex and $PR(\alpha_1, F) \subset PR(\alpha_2, F)$ for any $\alpha_2 \leq \alpha_1$ under some mild conditions. This implies that the deeper the contour, the more robust it is w.r.t. outliers in the data cloud (Ruts and Rousseeuw 1996). Thus, the depth contours can be used in practice to identify the outliers in a given data cloud.

As a special case, the innermost depth contour, which is a singleton in many situations, is the projection depth median (Zuo 2003)

$$PM(F) = PC\left(\alpha^*, F\right).$$

With appropriate choice of $\mu(F)$ and $\sigma(F)$ (see Remark 3.5 of Zuo 2003), $PM(F)$ can be shown to have the highest breakdown point among all the existing affine equivalent multivariate location estimators, much higher than that of the halfspace depth median.

Based on the above projection depth region $PR(\alpha, F)$, Zuo (2006) proposed the following projection depth trimmed mean (PTM)

$$PTM(\alpha, F) = \frac{\int_{PR(\alpha, F)} x w_1(PD(x, F)) F(dx)}{\int_{PR(\alpha, F)} w_1(PD(x, F)) F(dx)},$$

where $w_1(\cdot)$ is a suitable (bound) weight function on [0, 1]. As shown in Zuo (2006), PTM is highly robust locally as well as globally. When $\alpha = 0$, PTM degenerates into the famous Stahel-Donoho location estimator (Stahel 1981; Donoho and Gasko 1992), i.e. the projection weighted mean

$$PWM(F) = \frac{\int x w_1(PD(x, F)) F(dx)}{\int w_1(PD(x, F)) F(dx)}.$$

Besides of the location estimators, some other favorable scatter estimators can also be induced from the projection depth such as Stahel-Donoho scatter estimators (Zuo and Cui 2005),

$$PWS(F) = \frac{\int (x - PWM(F))(x - PWM(F))^T w_2(PD(x,F))F(dx)}{\int w_2(PD(x,F))F(dx)},$$

where $PWM(F)$ is the aforementioned Stahel-Donoho location estimator, $w_2(\cdot)$ denotes the weight function on $[0,1]$. To ensure $PTM(F)$, $PWM(F)$ and $PWS(F)$ to be well-defined, some regularity conditions are required, that is,

$$\int w_i\big(PD(x,F)\big)F(dx) > 0,$$

$$\int \|x\|^i w_i\big(PD(x,F)\big)F(dx) < \infty, \quad i = 1, 2.$$

With a finite sample $\mathcal{X}^n = \{X_1, X_2, \ldots, X_n\}$ in hand, the sample versions of the projection depth and its related estimators can be defined by simply replacing F with F_n, where F_n denotes the empirical distribution of F based on $\mathcal{X}^n$. Without confusion, in the following, we use $\mathcal{X}^n$ and F_n interchangeably.

3 The main idea

Note that the computations of $PD(x, \mathcal{X}^n)$, $PC(\alpha, \mathcal{X}^n)$, $PM(\mathcal{X}^n)$, $PTM(\alpha, \mathcal{X}^n)$, $PWM(\mathcal{X}^n)$ and $PWS(\mathcal{X}^n)$ depend mainly on that of $O(x, \mathcal{X}^n)$. Thus, let's first focus on the computation of $O(x, \mathcal{X}^n)$. In the sequel we assume that the data are in general position (Mosler et al. 2009), namely, every subset of $k+1$ data points generates an affine space of dimension k, $k = 1, 2, \ldots, p-1$. If the data are not in general position, the subsequent discussion and the algorithm need to be modified, e.g., by perturbing the data points by some random noise of a very small magnitude.

Since, with the choice of (Med, MAD), $Q(u, x, F)$ in (1) is odd w.r.t. u, we drop the absolute value symbol in $O(x, \mathcal{X}^n)$ and consider

$$O\big(x, \mathcal{X}^n\big) = \sup_{u \in \mathcal{S}^{p-1}} Q\big(u, x, \mathcal{X}^n\big)$$

instead, where

$$Q\big(u, x, \mathcal{X}^n\big) = \frac{u^T x - \mathrm{Med}(u^T \mathcal{X}^n)}{\mathrm{MAD}(u^T \mathcal{X}^n)},$$

$u^T \mathcal{X}^n = \{u^T X_1, u^T X_2, \ldots, u^T X_n\}$, and $\mathrm{Med}(\cdot)$ and $\mathrm{MAD}(\cdot)$ are defined in the sense that

$$\mathrm{Med}\big(\mathcal{Y}^n\big) = \frac{Y_{(\lfloor (n+1)/2 \rfloor)} + Y_{(\lfloor (n+2)/2 \rfloor)}}{2},$$

$$\mathrm{MAD}\big(\mathcal{Y}^n\big) = \mathrm{Med}\big\{\big|Y_i - \mathrm{Med}\big(Y^n\big)\big|, i = 1, 2, \ldots, n\big\},$$

where $\lfloor \cdot \rfloor$ is the floor function, and $Y_{(1)} \le Y_{(2)} \le \cdots \le Y_{(n)}$ denote the order statistics based on the univariate random variables $\mathcal{Y}^n = \{Y_1, Y_2, \ldots, Y_n\}$. For simplicity, hereafter we denote $m_1 = \lfloor (n+1)/2 \rfloor$ and $m_2 = \lfloor (n+2)/2 \rfloor$. Furthermore, since

$$\frac{u^T x - \mathrm{Med}(u^T \mathcal{X}^n)}{\mathrm{MAD}(u^T \mathcal{X}^n)} = \frac{w^T x - \mathrm{Med}(w^T \mathcal{X}^n)}{\mathrm{MAD}(w^T \mathcal{X}^n)}$$

holds for any $w = \lambda u$ with $\lambda > 0$, in what follows, we replace the constraint $u \in \mathcal{S}^n$ with $u \in R^p$ in $O(x, \mathcal{X}^n)$. In the sequel we do not eliminate the element $u = \mathbf{0}_p$ in the definition of $Q(u, x, \mathcal{X}^n)$, because when $u = \mathbf{0}_p$, $Q(u, x, \mathcal{X}^n) = 0 \le O(x, \mathcal{X}^n)$ always holds for any given x, where $\mathbf{0}_p$ denotes the p-dimensional zero vector.

Obviously, for a given data cloud $\mathcal{X}^n$, the difficulty of computing $O(x, \mathcal{X}^n)$ mainly comes from $\mathrm{Med}(u^T \mathcal{X}^n)$ and $\mathrm{MAD}(u^T \mathcal{X}^n)$. For convenience, denote $g(u) = \mathrm{Med}(u^T \mathcal{X}^n)$ and $h(u) = \mathrm{MAD}(u^T \mathcal{X}^n)$. Since $g(u)$ is involved in the definition of $h(u)$, let's now discuss the structure of $g(u)$.

Note that, for any given $u (\neq \mathbf{0}_p)$, there must exist a permutation, say $(i_1, i_2, \ldots, i_n)$, such that

$$u^T X_{i_1} \le u^T X_{i_2} \le \cdots \le u^T X_{i_n},$$

and the corresponding set $\mathcal{C} = \{t : \mathbb{A}_1^T t \le \mathbf{0}_{\kappa_1}\}$ is non-coplanar, where $\kappa_1 = n - 1$ and

$$\mathbb{A}_1 = (X_{i_1} - X_{i_{m_1}}, X_{i_2} - X_{i_{m_1}}, \ldots, X_{i_{m_1 - 1}} - X_{i_{m_1}}, X_{i_{m_1}} - X_{i_{m_1+1}}, \ldots, X_{i_{m_1}} - X_{i_n})$$

if n is odd, or $\kappa_1 = 2n - 3$ and $\mathbb{A}_1 = (\mathbb{M}_1, \mathbb{M}_2)$ with

$$\mathbb{M}_1 = (X_{i_1} - X_{i_{m_1}}, X_{i_2} - X_{i_{m_1}}, \ldots, X_{i_{m_1 - 1}} - X_{i_{m_1}}, X_{i_{m_1}} - X_{i_{m_2+1}}, \ldots, X_{i_{m_1}} - X_{i_n}),$$

$$\mathbb{M}_2 = (X_{i_1} - X_{i_{m_2}}, X_{i_2} - X_{i_{m_2}}, \ldots, X_{i_{m_1}} - X_{i_{m_2}}, X_{i_{m_2}} - X_{i_{m_2+1}}, \ldots, X_{i_{m_2}} - X_{i_n})$$

if n is even. Then some simple derivations can lead to that, for any $t \in \mathcal{C}$, we have

$$\begin{cases} X_{i_1}^T t \le X_{i_{m_1}}^T t \\ X_{i_2}^T t \le X_{i_{m_1}}^T t \\ \vdots \\ X_{i_{m_1-1}}^T t \le X_{i_{m_1}}^T t \\ X_{i_{m_1}}^T t \le X_{i_{m_1+1}}^T t \\ \vdots \\ X_{i_{m_1}}^T t \le X_{i_n}^T t \end{cases},$$

if n is odd (when n is even, the situation is similar). That is, for any $t \in \mathcal{C}$, it holds that $\mathrm{Med}(t^T \mathcal{X}^n) = t^T \mathrm{MedX}$, where MedX is fixed, i.e. independent of t, and $\mathrm{MedX} = (X_{i_{m_1}} + X_{i_{m_2}})/2$. This implies that $g(u)$ is essentially a piecewise linear function over R^p. Furthermore, for any $t \in \mathcal{C}$, since

$$X_{i_1}^T t, X_{i_2}^T t, \ldots, X_{i_{m_1-1}}^T t \\ \leq X_{i_{m_1}}^T t \leq \mathrm{Med}\big(t^T \mathcal{X}^n\big) \leq X_{i_{m_2}}^T t \leq X_{i_{m_2+1}}^T t, \ldots, X_{i_n}^T t,$$

we have

$$\big|t^T X_l - \mathrm{Med}\big(t^T \mathcal{X}^n\big)\big| \\ = \begin{cases} -(X_l - \mathrm{MedX})^T t, & \text{if } l \in \{i_1, i_2, \ldots, i_{m_1}\}, \\ (X_l - \mathrm{MedX})^T t, & \text{if } l \in \{i_{m_1+1}, i_{m_1+2}, \ldots, i_n\}, \end{cases}$$

$l = 1, 2, \ldots, n$. Namely, we could remove the absolute value symbol of $|t^T X_l - \mathrm{Med}(t^T \mathcal{X}^n)|$ according to the subscript l of X_l for any given $t \in \mathcal{C}$.

As to the function $h(u)$, similarly, we have that, for the given u and $(i_1, i_2, \ldots, i_n)$, there must exist an another permutation, say $(j_1, j_2, \ldots, j_n)$, such that

$$\big|u^T X_{j_1} - \mathrm{Med}\big(u^T \mathcal{X}^n\big)\big| \\ \leq \big|u^T X_{j_2} - \mathrm{Med}\big(u^T \mathcal{X}^n\big)\big| \leq \cdots \\ \leq \big|u^T X_{j_n} - \mathrm{Med}\big(u^T \mathcal{X}^n\big)\big|,$$

and the corresponding set $\mathcal{D} = \{t : \mathbb{A}^T t \leq \mathbf{0}_{\kappa_2}\}$ is non-coplanar, where $\kappa_2 = 2(n-1)$ and $\mathbb{A} = (\mathbb{A}_1, \mathbb{A}_2)$ with

$$\mathbb{A}_2 = (Z_{j_1} - Z_{j_{m_1}}, Z_{j_2} - Z_{j_{m_1}}, \ldots, Z_{j_{m_1-1}} - Z_{j_{m_1}}, Z_{j_{m_1}} \\ - Z_{j_{m_1+1}}, \ldots, Z_{j_{m_1}} - Z_{j_n}),$$

if n is odd, or $\kappa_2 = 4n-6$, $\mathbb{A} = (\mathbb{A}_1, \mathbb{A}_2)$ and $\mathbb{A}_2 = (\mathbb{N}_1, \mathbb{N}_2)$ with

$$\mathbb{N}_1 = (Z_{j_1} - Z_{j_{m_1}}, Z_{j_2} - Z_{j_{m_1}}, \ldots, Z_{j_{m_1-1}} - Z_{j_{m_1}}, Z_{j_{m_1}} \\ - Z_{j_{m_2+1}}, \ldots, Z_{j_{m_1}} - Z_{j_n}),$$

$$\mathbb{N}_2 = (Z_{j_1} - Z_{j_{m_2}}, Z_{j_2} - Z_{j_{m_2}}, \ldots, Z_{j_{m_1}} - Z_{j_{m_2}}, Z_{j_{m_2}} \\ - Z_{j_{m_2+1}}, \ldots, Z_{j_{m_2}} - Z_{j_n})$$

if n is even. Here $Z_{j_l} = s(j_l) \cdot (X_{j_l} - \mathrm{MedX})$, $l = 1, 2, \ldots, n$, with

$$s(i) = \begin{cases} -1, & \text{if } i \in \{i_1, i_2, \ldots, i_{m_1}\} \\ 1, & \text{if } i \in \{i_{m_1+1}, i_{m_1+2}, \ldots, i_n\} \end{cases}.$$

Obviously, $\mathcal{D} \subset \mathcal{C}$, and for any $t \in \mathcal{D}$, we have that $\mathrm{MAD}(t^T \mathcal{X}^n) = t^T \mathrm{MADX}$, where MADX is fixed, i.e. independent of t, and $\mathrm{MADX} = (Z_{j_{m_1}} + Z_{j_{m_2}})/2$. This implies that $h(u)$ is also a piecewise linear function over R^p.

So far, we actually obtain that $Q(u, x, \mathcal{X}^n) = c^T u / d^T u$ over $\mathcal{D}$ w.r.t. u, where $c = x - \mathrm{MedX}$ and $d = \mathrm{MADX}$. That is, for a given data cloud $\mathcal{X}^n$, $Q(u, x, \mathcal{X}^n)$ is in essence a piecewise linear fractional function w.r.t. $u \in R^p$ over a finite number of pieces such as $\mathcal{D}$. Then, similar to Liu et al. (2011), by using the theory of linear fractional functionals programming (Swarup 1965), we can show that, within each $\mathcal{D}$, the maximum of $Q(u, x, \mathcal{X}^n)$ occurs only at the edges, i.e. one-dimensional facets, of $\mathcal{D}$ for $p \geq 2$ (when $p = 2$, see also Zuo and Lai 2011).

Therefore, to find the global maximum value of $Q(u, x, \mathcal{X}^n)$, it is sufficient to calculate the local maximum values over each piece at first, and then find the global maximum value from all of these local maximum values. This implies that the task of computing $O(x, \mathcal{X}^n)$ can be divided into two steps: (1) find all the possible cones $\mathcal{D}$, (2) for each given $\mathcal{D}$, obtain all of its edges.

Typically, each $\mathcal{D}$ forms a polyhedral cone. Each facet $\mathcal{F}$ of $\mathcal{D}$ corresponds to a non-redundant constraint in $\mathbb{A}^T t \leq \mathbf{0}_{\kappa_2}$. All of these cones together span the whole space R^p. Note that, there must be one and only one facet shared between every two adjacent cones. This is why one may simply pass through all the cones counter-clockwise when $p = 2$ (Zuo and Lai 2011). For $p > 2$, the problem is far more complicated. However, it is still possible to utilize the breadth-first search algorithm to fulfill this task. See for example Sect. 2.3 of Paindaveine and Šiman (2012a) for a similar discussion.

Note that, for any given cone $\mathcal{D}$, all of its facets $\mathcal{F}$ can be identified uniquely by the mean of all the vertices of $\mathcal{F} \cap V$, where $V = [-1, 1]^p$. That is, instead of $\mathcal{D}$, we could consider the polytope $\mathcal{D} \cap V$, i.e. $\{t : \mathbb{B}^T t \leq \mathbf{b}\}$, where

$$\mathbb{B} = (\mathbb{A}, \mathbb{I}_p, -\mathbb{I}_p), \quad \text{and} \quad \mathbf{b} = \big(\mathbf{0}_{\kappa_2}^T, \mathbf{1}_{2p}^T\big)^T$$

with $\mathbb{I}_l$ being an $l \times l$ identity matrix, $\mathbf{1}_l$ bing an l-dimensional vector of ones. Then, similar to Paindaveine and Šiman (2012a), the program *qhull* (Barber et al. 1996) can be employed to find all the vertices and facets. In Matlab, a similar task can be fulfilled by *con2vert.m*, which is developed by Michael Kleder and can be downloaded from Matlab Central File Exchange, by means of the dual relationship between vertex and facet enumeration (Bremner et al. 1998).

This actually completes the second step. Now let us focus on the task of how to find all the possible cones $\mathcal{D}$. Note that, to obtain $\mathcal{D}$, one has to compute MedX and remove the absolute value symbols of $|t^T X_l - \mathrm{Med}(t^T \mathcal{X}^n)|$ $(l = 1, 2, \ldots, n)$ in advance. Then, similar to Zuo and Lai (2011), we:

- Firstly divide the whole space R^p into a serial of direction vector regions $\mathcal{C}$ such that the median of the projected data within each region $\mathcal{C}$ will be the projection of a fixed data point (when n is odd) or the middle point of a line segment connecting two data points (when n is even), namely, both (i_{m_1}, i_{m_2}) and MedX are fixed within each given $\mathcal{C}$, and the absolute value symbols of $|t^T X_l - \mathrm{Med}(t^T \mathcal{X}^n)|$ ($l = 1, 2, \dots, n$) can be conveniently removed for any $t \in \mathcal{C}$;

- For each given $\mathcal{C}$, further divide it into a serial of direction vector regions $\mathcal{D}$ such that the MAD of the projected data within each region $\mathcal{D}$ will only depend on the absolute value(s) of the projection of one (or two if n is even) fixed original data point(s).

Remark 1 In practice, there may exist some other ways to divide the space R^p (when $p = 2$, see for example Zuo and Lai 2011) into a serial of direction vector regions such that, within each regions, the function $g(u)$ is simple a linear function. However, the way of using some cones such as $\mathcal{C}$ has many desirable properties. For example, during the computation, there is no need to take care of the ordering existing in both $t^T X_{i_1}, t^T X_{i_2}, \dots, t^T X_{i_{m_1 - 1}}$ and $t^T X_{i_{m_2+1}}, t^T X_{i_{m_2+2}}, \dots, t^T X_{i_n}$, since $x_{i_1}, x_{i_2}, \dots, x_{i_{m_1-1}} \le x_{i_{m_1}}, x_{i_{m_2}} \le x_{i_{m_2+1}}, \dots, x_{i_n}$ can also guarantee $x = (x_{i_{m_1}} + x_{i_{m_2}})/2$ to be the median of $x_1, x_2, \dots, x_n$ (Floyd and Rivest 1975). As a result, the computation can become more efficient. Furthermore, by using $\mathcal{C}$, it is very convenient to remove the difficulty caused by the absolute value symbols involved in $\mathrm{MAD}(u^T \mathcal{X}^n)$.

Finally, the optimal direction vectors are then the unit vectors along the direction of the edges of $\mathcal{D}$. Without loss of generality, denote $\mathcal{U} = \{u_i\}_{i=1}^M$ to be the optimal direction vectors corresponding to $\mathcal{X}^n$, where M denotes the number of u_i's. Then similar to Liu et al. (2011), for any $x \in R^p$, we have

$$O(x, \mathcal{X}^n) = \max_{1 \le i \le M} \hbar_i(x), \tag{2}$$

where $\hbar_i(x) = \mathbf{a}_i^T x + \mathbf{b}_i$ with $\mathbf{a}_i = u_i / \mathrm{MAD}(u_i^T \mathcal{X}^n)$ and $\mathbf{b}_i = -\mathrm{Med}(u_i^T \mathcal{X}^n)/\mathrm{MAD}(u_i^T \mathcal{X}^n)$. Based on (2), $PD(x, \mathcal{X}^n)$, $PTM(\alpha, \mathcal{X}^n)$, $PWM(\mathcal{X}^n)$ and $PWS(\mathcal{X}^n)$ can be computed directly according to their definition. For $PC(\alpha, \mathcal{X}^n)$, $PM(\mathcal{X}^n)$, see Liu et al. (2011) for a detailed discussion.

4 Algorithm

Since the optimal direction vectors play a key roll in the computation of the projection depth and its associated estimators, we will propose an algorithm to fulfill the task of finding all the optimal direction vectors in this section. The detailed procedure is listed as follows.

1.1. Set $\mathcal{A}_{old} = \emptyset$, $\mathcal{U}_{old} = \emptyset$, and $\mathcal{T}_{old} = \emptyset$.

1.2. Generate a random unit vector u_0 repeatedly until it satisfies $u_0^T X_{i_1} < u_0^T X_{i_2} < \cdots < u_0^T X_{i_n}$. Store the corresponding permutation $(i_1, i_2, \dots, i_n)$.

1.3. Based on $(i_1, i_2, \dots, i_n)$, compute $\mathrm{MedX} = (X_{i_{m_1}} + X_{i_{m_2}})/2$ and the matrix $\mathbb{A}_1$. Find all the vertices and facets of $\mathcal{C} \cap V$, i.e. $\{t : \mathbb{B}_1^T t \le \mathbf{b}_1\}$, where $\mathbb{B}_1 = (\mathbb{A}_1, \mathbb{I}_p, \mathbb{I}_p)$ and $\mathbf{b}_1 = (0_{\kappa_1}^T, \mathbf{1}_{2p}^T)^T$. Drop the facets not being the constraints of $\mathcal{C}$. For each remaining facet $\mathcal{F}$, compute its $u_{\mathcal{F}}$ and $\mathcal{I}_{\mathcal{F}}$. Assign all the $u_{\mathcal{F}}$ and $(u_{\mathcal{F}}, \mathcal{I}_{\mathcal{F}})$ to $\mathcal{A}_{new}$ and $\mathcal{T}_{new}$, respectively. Store the coefficient matrix of the non-redundant constraints of $\mathcal{C}$ in $\widetilde{\mathbb{A}}_1$. Therefore, it also holds that $\mathcal{C} = \{t \in R^p : \widetilde{\mathbb{A}}_1^T t \le \mathbf{0}_{\widetilde{\kappa}_1}\}$, where $\widetilde{\kappa}_1$ denotes the number of the non-redundant constraints in $\mathcal{C}$.

In the above, $u_{\mathcal{F}}$ is defined to be the mean of all the vertices of $\mathcal{F} \cap V$, and $\mathcal{I}_{\mathcal{F}}$ denotes the subscript pairs (from left to right) of the constraints of $\mathbb{A}_1^T t \le \mathbf{0}_{\kappa_1}$ that correspond to $\mathcal{F}$, for example, if $\mathcal{F}$ corresponds to the first constraint, then we have $\mathcal{I}_{\mathcal{F}} = \{(i_1, i_{m_1})\}$ (if $\mathcal{F}$ corresponds to more than one constraints, such as the first two constraints of $\mathbb{A}_1^T t \le \mathbf{0}_{\kappa_1}$, then $\mathcal{I}_{\mathcal{F}}$ contains more than one subscript pair, i.e. $\mathcal{I}_{\mathcal{F}} = \{(i_1, i_{m_1}), (i_2, i_{m_1})\}$).

For more detailed discussion about how to find the non-redundant constraints, facets and interior points, see for example the Technical details listed in the Appendix of Paindaveine and Šiman (2012a).

1.4. Obtain all the optimal direction vectors existing in $\mathcal{C}$. Store them into $\mathcal{U}_{new}$. The detailed procedure can be described as follows.

(a) Let $\mathcal{B}_{old} = \emptyset$, $\Lambda_{old} = \emptyset$, and $\mathcal{U}_{new} = \emptyset$.

(b) Compute $\mathcal{Z}^n = \{Z_1, Z_2, \dots, Z_n\}$, where $Z_l = s(l) \cdot (X_l - \mathrm{MedX})$, $l = 1, 2, \dots, n$. Set $v_0 = u_0$ and obtain the permutation $(j_1, j_2, \dots, j_n)$ corresponding to the projected data $v_0^T \mathcal{Z}^n = \{v_0^T Z_1, v_0^T Z_2, \dots, v_0^T Z_n\}$.

(c) For the permutation $(j_1, j_2, \dots, j_n)$, compute $\mathrm{MADX} = (Z_{j_{m_1}} + Z_{j_{m_2}})/2$ and the matrix $\mathbb{A}_2$. Find all the vertices and facets of $\widetilde{\mathcal{D}} \cap V$, where $\widetilde{\mathcal{D}} = \{t : \widetilde{\mathbb{A}}^T t \le \mathbf{0}_{\widetilde{\kappa}_1 + \kappa_1}\}$ with $\widetilde{\mathbb{A}} = (\widetilde{\mathbb{A}}_1, \mathbb{A}_2)$. Drop the facets not being the constraints of $\mathbb{A}_2^T t \le \mathbf{0}_{\kappa_1}$. For each remaining facet $\bar{\mathcal{F}}$, compute its corresponding $u_{\bar{\mathcal{F}}}$ and $\mathcal{I}_{\bar{\mathcal{F}}}$. Assign all the $u_{\bar{\mathcal{F}}}$ and $(u_{\bar{\mathcal{F}}}, \mathcal{I}_{\bar{\mathcal{F}}})$ to $\mathcal{B}_{new}$ and Λ_{new}, respectively.

(d) Standardize all the vertices of $\widetilde{\mathcal{D}}$ into the unit vectors, and assign them to $\mathcal{U}_{tmp}$. For each $u \in \mathcal{U}_{tmp}$, check whether it exists in $\mathcal{U}_{new}$. If it does, do nothing, otherwise, add u into $\mathcal{U}_{new}$, and obtain its corresponding Med value, i.e. $u^T\mathrm{MedX}$ and MAD value, i.e. $u^T\mathrm{MADX}$.

(e) Based on $\mathcal{B}_{new}$ and Λ_{new}, update $\mathcal{B}_{old}$ and Λ_{old} by using the following procedure. Set $\mathcal{B}_{temp} = \emptyset$, $\Lambda_{temp}^1 = \emptyset$, and $\Lambda_{temp}^2 = \emptyset$. For every element $u_{\bar{\mathcal{F}}}$

of $\mathcal{B}_{new}$, check whether it exists in $\mathcal{B}_{old}$. If it does, add $(u_{\bar{\mathcal{F}}}, \mathcal{I}_{\bar{\mathcal{F}}})$ into the set Λ^1_{temp}, otherwise, add $u_{\bar{\mathcal{F}}}$ and $(u_{\bar{\mathcal{F}}}, \mathcal{I}_{\bar{\mathcal{F}}})$ into the sets $\mathcal{B}_{temp}$ and Λ^2_{temp}, respectively (in fact, here Λ^1_{temp} contains the facets that have been considered, while Λ^2_{temp} contains those unconsidered). Update $\mathcal{B}_{old}$ by adding all the elements $u_{\bar{\mathcal{F}}}$ of $\mathcal{B}_{temp}$ into $\mathcal{B}_{old}$. Update Λ_{old} by first eliminating the facets that exist in both Λ_{old} and Λ^1_{temp} from Λ_{old}, and then adding the facets of Λ^2_{temp} into Λ_{old}.

(f) Check whether Λ_{old} is empty. If it is, terminate Step 1.4, and proceed to the next step. If not, obtain a new inner point v_0 of the cone that is adjacent to the first facet $\bar{\mathcal{F}}$ in Λ_{old} based on $(u_{\bar{\mathcal{F}}}, \mathcal{I}_{\bar{\mathcal{F}}})$. Here a good candidate of v_0 may be $u_{\bar{\mathcal{F}}} + \lambda \times (Z_{j_{k'_1}} - Z_{j_{k'_2}})$, where $(j_{k'_1}, j_{k'_2})$ denotes the first pair in $\mathcal{I}_{\bar{\mathcal{F}}}$, and λ is a very small positive magnitude such as 10^{-10}.

(g) Check whether v_0 satisfies $\widetilde{\mathbb{A}}_1^T v_0 \leq \mathbf{0}_{\widetilde{\kappa}_1}$. If it does not (this means that $v_0 \notin \mathcal{C}$ and the current facet $\bar{\mathcal{F}}$ is already the boundary of $\mathcal{C}$), then remove $\bar{\mathcal{F}} := (u_{\bar{\mathcal{F}}}, \mathcal{I}_{\bar{\mathcal{F}}})$ from Λ_{old}. Go back to Step 1.4-(f). If it does, obtain the permutation corresponding to v_0. Go back to Step 1.4-(c).

1.5. Based on $\mathcal{A}_{new}$ and $\mathcal{T}_{new}$, update $\mathcal{A}_{old}$ and $\mathcal{T}_{old}$ by using a similar method as in Step 1.4-(e) for updating $\mathcal{B}_{old}$ and Λ_{old}. For each direction vector u of $\mathcal{U}_{new}$, check whether it exists in $\mathcal{U}_{old}$. If it does, do nothing, otherwise, add it into $\mathcal{U}_{old}$.

1.6. Check whether $\mathcal{T}_{old}$ is empty. If it is, terminate the algorithm successfully. If not, similar to Step 1.4-(f), obtain a new inner point u_0 based on the first element $(u_{\mathcal{F}_1}, \mathcal{I}_{\mathcal{F}_1})$ of $\mathcal{T}_{old}$. Compute u_0's permutation. Go back to Step 1.3.

Finally, $\mathcal{U}_{old}$ is what we need. For this algorithm, we have an implementation by using Matlab. The corresponding codes can be obtained from the corresponding author (zuo@msu.edu).

Remark 2 Note that the permutation obtained from the projections of $\mathcal{X}^n$ onto u is exactly the reverse of the permutation of the projections of $\mathcal{X}^n$ onto $-u$. Therefore, in practice, it is sufficient to find the direction vectors, say $\{v_i\}_{i=1}^{M_1}$, in one halfspace with the origin on its boundary, such as $\{t : \mathbf{e}_{p1}^T z \geq 0\}$, and then define the final optimal direction vectors as $\{u_i\}_{i=1}^M = \{v_1, -v_1, \ldots, v_{M_1}, -v_{M_1}\}$, where $M = 2M_1$, and $\mathbf{e}_{p1} = (1, 0, \ldots, 0)^T$ is a p-vector.

5 Examples

In order to gain more insight into the performance of the proposed algorithm, some numerical examples are provided in this section based on the Matlab implementation. Both real and simulated data are considered here.

5.1 Real data

We start with a real data set, which is taken from the Table 9 of Rousseeuw and Leroy (1987) (page 94) and consists of 75 observations. See also Hawkins et al. (1984). The goal here is not to perform a thorough analysis for data, but rather to show how the algorithm performs and how the three-dimensional projection depth contours as well as other projection depth related estimators look like in practice.

To ensure the inputed data are in general position, a robust standardization, as well as a random perturbation, of the data is performed before the actual computation. That is, we first have each component x_{ij} centered around its median and divided by its MAD, i.e.

$$x_{ij} = \frac{x_{ij} - \mathrm{Med}(\mathcal{X}_j^n)}{\mathrm{MAD}(\mathcal{X}_j^n)},$$

where $\mathcal{X}_j^n = \{x_{1j}, x_{2j}, \ldots, x_{nj}\}$ with $n = 75$, $1 \leq i \leq 75$ and $1 \leq j \leq 3$, and then further perturb these data points by some random noises of a very small magnitude, say $e \times 10^{-6}$, where $e \sim N(0, 1)$. The resulting data are listed in Table 1. Its corresponding scatter plot is given in Fig. 1(a). From this figure, we can see that there are a few outliers, far away from the majority of the data. In fact, these outliers correspond to the data points 1–14 of Table 1.

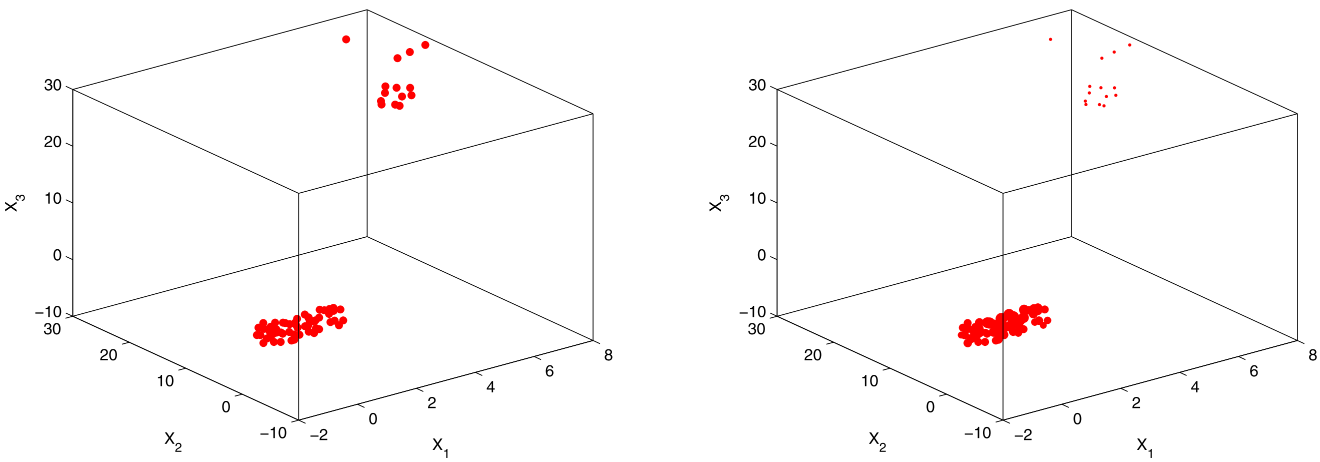
Based on the proposed algorithm, a finite number of optimal direction vectors can be obtained. The exact projection depth values of the sample points w.r.t. the data cloud $\mathcal{X}^n$ are reported in Table 2. For the sake of comparison, we also compute the approximate projection depth values based on 5×10^5 random direction vectors. From Table 2, we can see that, the exact projection depth values are smaller than all of the approximate ones even when the number of random direction vectors is 5×10^5. This indicates that our algorithm can successfully find the optimal direction vectors. Actually, to ensure this result, one can generate a larger number of direction vectors, and then check whether the approximate values based on these vectors are smaller than our exact ones.

Furthermore, we also plot the projection depth-size plot of this data set in Fig. 1(b), where the bigger the points, the larger the depth values are. From this figure, we can see that, the depth of the central points are lager than those on the skirts. This confirms that the projection depth can provide a center-outward ordering for a given data cloud.

As discussed in Liu et al. (2011), the optimal direction vectors are only dependent on the data cloud, i.e. x-free. Therefore, with these vectors in hand, we are able to

Table 1 The transformed Artificial Data Set of Hawkins (Rousseeuw and Leroy 1987)

Index	x_1	x_2	x_3	Index	x_1	x_2	x_3	Index	x_1	x_2	x_3
1	6.3846136693	15.8181809041	21.8333332224	26	−0.6923073768	0.9999999774	0.3333330953	51	0.3846133339	−0.6363648998	−1.4166670038
2	5.9230769631	16.6363641999	22.3333339156	27	1.1538455284	0.2727263750	0.6666676285	52	1.1538456052	−1.4545443932	−0.7499999033
3	6.8461550107	16.3636360594	24.0833351753	28	0.0000009377	−1.2727270982	−0.0833318794	53	−1.1538454727	−1.6363623755	0.9999996725
4	6.2307702594	17.5454536551	24.6666670032	29	−0.4615387598	−1.1818172041	−1.0833330684	54	−0.5384617882	0.7272715978	−1.4999987027
5	6.5384617900	17.1818190931	24.1666668181	30	−0.4615395215	−1.3636352729	1.0833331987	55	−0.9999996980	0.1818190090	−1.0000009497
6	6.9230770856	16.5454555306	22.5833330938	31	0.9999988103	−0.7272730831	−0.9166662745	56	0.0000003305	0.9090908421	−0.9999982830
7	6.6923068173	17.0000004971	22.4999985490	32	−1.0000014041	0.1818190183	−1.5000007347	57	−0.0000003352	−1.3636367583	−1.1666681752
8	6.2307690208	15.8181829220	22.2499988454	33	−0.2307689861	0.8181808307	−0.4999998544	58	0.4615391751	1.0909083068	−0.4999984965
9	6.0769216872	16.8181817815	24.0833351311	34	−1.0769225956	−2.0000000178	−1.1666651660	59	−0.1538476821	−0.0909090043	0.7500005065
10	5.7692327700	15.9090916136	23.5000002067	35	0.9999997911	0.1818192372	0.7500006171	60	−1.1538458773	−0.6363623566	1.0000005432
11	7.0769227520	19.8181824360	27.4166677131	36	−0.5384614225	0.0000006631	0.5000002324	61	−1.0769234278	1.0909087261	0.7499985909
12	7.8461542088	18.9090911762	29.0833338447	37	−1.3076937527	0.7272737103	0.4166665325	62	−0.6923082974	−1.9090907051	−1.5000017421
13	7.8461556499	21.6363621584	26.5833339289	38	−0.2307685199	−0.9090914314	−1.5833332439	63	−0.5384617678	0.4545457345	−1.5833352699
14	7.0769237266	28.9090929561	26.5833337155	39	0.2307685451	−2.0000013706	−0.7500014669	64	0.7692308109	0.7272717290	0.6666679310
15	1.2307696085	0.6363631120	0.0000002511	40	−0.9999990698	−0.1818182099	−0.7499997559	65	0.1538462091	−1.3636336183	0.4999999124
16	1.0000002901	−0.0000002770	−1.4999989714	41	1.2307696755	−0.5454553025	0.6666670785	66	−1.2307682431	−0.3636344556	−1.0833336459
17	−1.3846170424	−0.5454536306	−1.5833343856	42	−1.1538465191	−1.0909073022	0.4999996406	67	−0.1538461050	−0.1818188507	−0.7499999417
18	0.3846158772	−0.5454534334	−0.0833321672	43	−1.3076940463	1.0000012241	−1.0000002636	68	−1.3076929212	−1.9999995981	−0.8333335847
19	−0.7692291927	0.6363638897	−0.4166651231	44	−0.0000003606	−1.5454528102	0.9166653956	69	0.1538456874	−1.4545469557	−1.4999996476
20	0.9999995450	1.0909082446	0.0833343420	45	0.0769221398	−1.9090888643	−1.2500007502	70	−0.6153854960	−0.0000014284	0.6666676661
21	0.6153839126	0.0000012352	−0.1666649134	46	0.0000005645	−1.5454547522	0.7499997281	71	0.3076942232	0.2727272806	0.1666663896
22	−1.0769211223	0.9090911404	−0.1666659949	47	0.9230761750	−1.9090908736	−1.0833317440	72	−0.9230779071	−0.1818163775	−0.5000015540
23	0.1538455901	0.0909107927	−1.0833333689	48	1.0000013151	−0.5454530925	0.7500008459	73	−1.1538450973	−0.4545442840	0.0833329945
24	−0.3846155572	0.0909069591	−1.3333334688	49	0.9999970526	0.2727288062	−0.1666675493	74	−1.3846157836	0.0000012303	−0.4166660703
25	−0.6153865708	−2.0000000663	−1.4166652857	50	0.2307673566	0.5454564788	0.6666661890	75	−1.1538472341	−1.6363636702	0.4166676097



(a) The scatter plot the data points

(b) The projection depth-size plot of the data points

Fig. 1 The scatter plot and the projection depth-size plot of the data points

compute all the associated estimators, including the Stahel-Donoho estimators, projection trimmed mean, projection median (Table 3). Their corresponding scatter plots are given in Fig. 2. From this figure, it is easy to see that the ordinary mean is dragged outside the bulk of the data by a few outliers (Fig. 2(a)), while the other projection depth based estimators are all located among the majority of the data (Figs. 2(b)–2(d)). See also their depth values and their corresponding rank reported in Table 3.

In addition, we also provide the ordinary covariance matrix and projection depth weighted scatter estimator as follows.

$$M_1 = \begin{pmatrix} 7.8945 & 19.9085 & 26.4385 \\ 19.9085 & 56.1016 & 71.7164 \\ 26.4385 & 71.7164 & 95.7187 \end{pmatrix},$$

$$M_2 = \begin{pmatrix} 0.6699 & 0.0355 & 0.0752 \\ 0.0355 & 0.9523 & 0.1066 \\ 0.0752 & 0.1066 & 0.7432 \end{pmatrix},$$

$$M_3 = \begin{pmatrix} 0.7214 & 0.2085 & 0.3126 \\ 0.2085 & 1.3985 & 0.7357 \\ 0.3126 & 0.7357 & 1.5689 \end{pmatrix},$$

where M_1 denotes the ordinary covariance matrix based on all the data, M_2 denotes the ordinary covariance matrix based on the data points 15–75, and M_3 denotes the projection depth weighted scatter estimator. From these three matrices, we can see that the ordinary covariance matrix is impacted significantly by the outliers. On the other hand, the impact of the outliers on M_3 is very limited. This, together with the discussion about the location estimators, confirms the high robustness of the projection depth and its associated estimators (Zuo 2003, 2006). It is noteworthy that, during the computation of *PWM*, *PTM* and *PWS*, the weight functions $w_i(\cdot)$, $i = 1, 2$, we used here are taken from Zuo and Cui (2005) (see (4) of page 385) with $K = 3$ and C being the median of all the projection depth values.

Finally, we also provide the projection depth contours in Fig. 3. From Fig. 3, we can see that, similar to the halfspace depth contours (Paindaveine and Šiman 2012a), the projection depth contours are also polyhedral.

5.2 Simulated data

Like in many other algorithms for computing multivariate depths, a commonly concerned issue is the computational time complexity (CTC) of the proposed algorithm. Obviously, the CTC here depends on the number of the computational loops, which in turn depends on the number of the cones $\mathcal{D}$. Similar to Paindaveine and Šiman (2012a), if the average number of the cones $\mathcal{D}$ is denoted to be $N(n, p)$, the average computational complexity of the algorithm amounts to $O(n \log(n) N(n, p))$. However, in practice, $N(n, p)$ depends on the specific data configuration, and it is difficult to obtain its specific expression.

In this subsection, some empirical results are provided based on some simulated data to examine the CTC of a Matlab implementation of the proposed algorithm. All of these results are obtained on a Dell inspiron 1525 laptop with Intel(R) Pentium(R) Dual 2.00 GHz, RAM 2.00 GB, Windows Vista™ Home Basic and Matlab 7.8. Of course, the empirical results would be different in other different hardware or software settings.

The data are generated from the p-dimensional standard normal distribution $N(\mathbf{0}_p, \mathbb{I}_p)$, where $p = 2, 3$. For each dimension, the sample sizes are taken to be $n = 40, 80, 160, 320$, respectively. For each sample size n, the

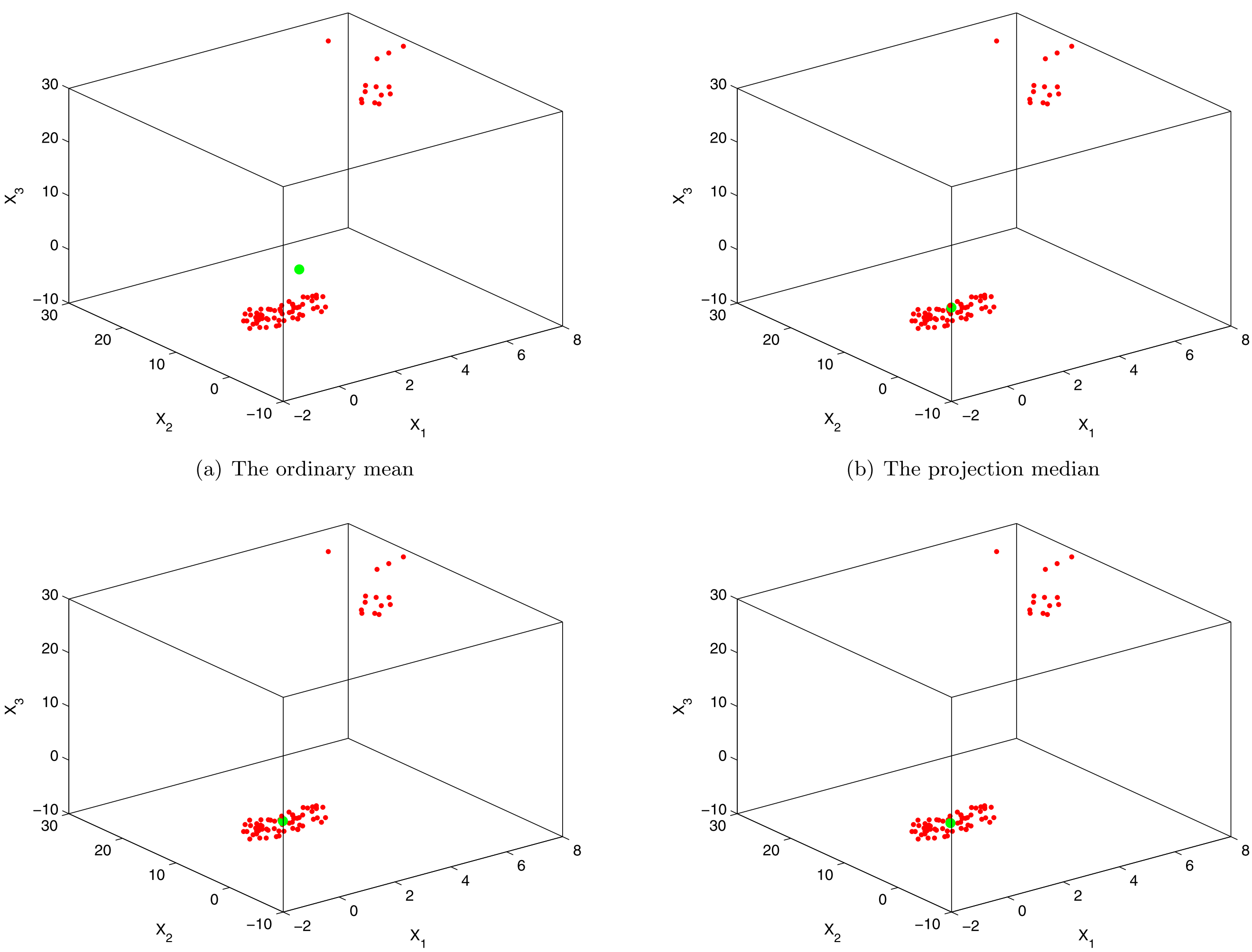
Table 2 The computed projection depth values

Index	ExPDV	<	RndPDV(500000)	Index	ExPDV	<	RndPDV(500000)
1	0.026971091596813	1	0.027010295179358	39	0.242858287786391	1	0.243543195324487
2	0.025940274319797	1	0.025977883561525	40	0.328193986268838	1	0.329356398273369
3	0.025156404707486	1	0.025192695081099	41	0.255847342471990	1	0.256900260339055
4	0.023967514509570	1	0.024002029014211	42	0.236937127261457	1	0.237090298335356
5	0.024536491876213	1	0.024571950782570	43	0.193444290339650	1	0.194091499971123
6	0.026044418345459	1	0.026082434984171	44	0.228084150667704	1	0.228897745063334
7	0.025745396855716	1	0.025783022211722	45	0.230865216609180	1	0.231608221576251
8	0.026634153111971	1	0.026672699520511	46	0.237288304933460	1	0.238115457712683
9	0.024721264053494	1	0.024756789275117	47	0.208146282345113	1	0.208528903329065
10	0.025611879434366	1	0.025648508524182	48	0.281632264453516	1	0.282795601604059
11	0.021496473007716	1	0.021527533156543	49	0.288081592208114	1	0.290655568430341
12	0.021256468931249	1	0.021286932845690	50	0.400060334153352	1	0.400381469641337
13	0.021157187858503	1	0.021188355071626	51	0.291118646713453	1	0.291468191783047
14	0.018239676615890	1	0.018267196526506	52	0.225792127873946	1	0.226596161494754
15	0.256557707578147	1	0.260449385882599	53	0.182602140289277	1	0.182756096448804
16	0.226292737637159	1	0.227882602979972	54	0.230132188958697	1	0.230890399950662
17	0.279908427070492	1	0.281102421899932	55	0.274903032424768	1	0.275432301503263
18	0.378063869311529	1	0.379078309343569	56	0.270539108572681	1	0.270874312401234
19	0.278588657767958	1	0.279542468279088	57	0.279144377456144	1	0.279976934348983
20	0.277699080038189	1	0.281334463818740	58	0.269893934893001	1	0.270510768068535
21	0.369399859811068	1	0.371454437385178	59	0.426188361094133	1	0.426679373731618
22	0.242025428727202	1	0.243414625058031	60	0.236220260971516	1	0.238051768030542
23	0.352298689039827	1	0.354594014506247	61	0.257371139687842	1	0.258224411364075
24	0.321096374677087	1	0.321996833054720	62	0.241233853130350	1	0.241603787787572
25	0.239909015806497	1	0.240287191429531	63	0.249230138307258	1	0.249955321573323
26	0.289067442513903	1	0.290641794303457	64	0.339085146413927	1	0.339963109249454
27	0.304674639187652	1	0.305149243282453	65	0.271468579720687	1	0.272263235733522
28	0.335999996865132	1	0.336987233804754	66	0.301416904380780	1	0.302476099262877
29	0.328395422443053	1	0.328810124469507	67	0.499999999999994	1	0.500000000000000
30	0.216440956141821	1	0.216608430686738	68	0.246479072870450	1	0.246639230386372
31	0.265425232904363	1	0.266043306066031	69	0.244840404383918	1	0.245620312034946
32	0.247600879549658	1	0.248035479348443	70	0.373246018257127	1	0.373579100512335
33	0.298715696425150	1	0.300124298379311	71	0.486600061858371	1	0.487541043046627
34	0.248384988633959	1	0.248746519013890	72	0.355596597428383	1	0.356504008397354
35	0.323366517828188	1	0.324472485469382	73	0.305612882880774	1	0.305899628739744
36	0.408556400377977	1	0.408874859094819	74	0.275647098316914	1	0.276482848171312
37	0.252738737895226	1	0.253490820374786	75	0.206932300492047	1	0.207059688945820
38	0.300993544478040	1	0.301223140008111				

computation was run 100 times repeatedly. We report the average execution times of computing all the depth values of $\mathcal{X}^n$ based on the optimal direction vectors, M_r random direction vectors (random method) and M_f fixed direction vectors (fixed method), respectively, in Table 4; see the columns named AET. Here the random directions are uniformly distributed on $\mathcal{S}^{p-1}$, and the fixed directions are generated by using a grid search method. $M_r = M_f = 2000$ when $p = 2$ and $M_r = M_f = 5 \times 10^5$ when $p = 3$. M and L denote the average number (approximately) of the optimal direction vectors and the cones $\mathcal{D}$, respectively. For the sake of comparison, we also compute the empirical mean abso-

Table 3 The ordinary mean and the projection depth based location estimators, where the column ExPDV contains the corresponding depth values, and the column Rank denotes the ranks of the estimators among the data in term of the depth value. Here Rank $= k$ means that there are $k-1$ data points having their depth values greater than that of the estimator

Estimators	x_1	x_2	x_3	ExPDV	Rank
Mean	1.0821	3.0885	4.2756	0.121717825301521	62
PM	−0.0810	0.0405	0.2084	0.636655972019341	1
PWM	−0.1367	−0.2139	−0.1356	0.604832356541257	1
PTM (0.05)	−0.1958	−0.3717	−0.3482	0.598872703877245	1



(a) The ordinary mean

(b) The projection median

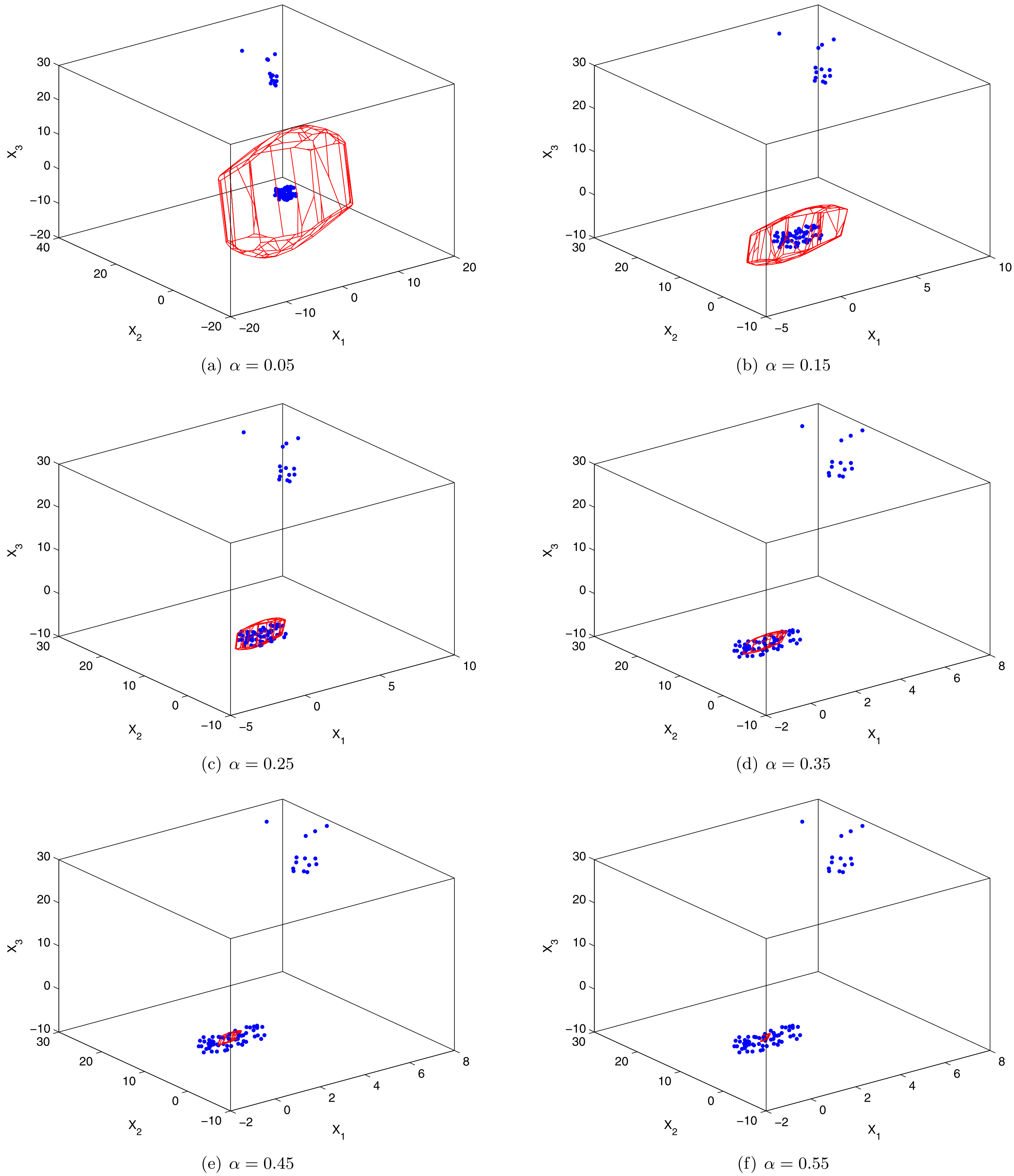
(c) The projection weighted mean, i.e. Stahel-Donoho location estimator

(d) The projection trimmed mean with $\alpha = 0.05$

Fig. 2 The ordinary mean and some location estimators based on projection depth

lute error of these three methods: $\text{EMAE} = \frac{1}{R}\sum_{i=1}^{R} \text{EAE}_i$ and $\text{EAE}_i = \sum_{i=1}^{n} |PD(X_i, \mathcal{X}^n) - APD(X_i, \mathcal{X}^n)|$, where $R = 500$, and $APD(X_i, \mathcal{X}^n)$ denotes the approximate depth value of X_i based on the random or fixed direction vectors. Based on Table 4, it seems reasonable to assume the CTC of the proposed algorithm to be not worse than $O(n^p)$ on average. Furthermore, it seems that the random method performs better than the fixed method in terms of both execution time and accuracy.

As a special case, when $p = 2$, the algorithm proposed in this paper is very efficient, and much more efficient than that of Zuo and Lai (2011). Figure 4 reports a compari-



(a) $\alpha = 0.05$ (b) $\alpha = 0.15$

(c) $\alpha = 0.25$ (d) $\alpha = 0.35$

(e) $\alpha = 0.45$ (f) $\alpha = 0.55$

Fig. 3 The projection depth contours, where Figs. (**a**)–(**d**) correspond to $\alpha = 0.05, 0.15, \ldots, 0.55$, respectively

son between the proposed procedure and the existing one of Zuo and Lai (2011). From this table, it is easy to see that the proposed procedure is much faster than that of Zuo and Lai (2011). In fact, a lot of numerical experiments indicate that it takes only few seconds by using the proposed algorithm to compute all the depth values of a given data cloud even when n is large as 1000. See Fig. 5 for the average execution time of the proposed algorithm based on 100

Table 4 Average execution times (in second) of different algorithms and the empirical mean absolute error (EMAE)

p	$n \backslash$	Exact				Random		Fixed	
		AET	EMAE	$M(\approx)$	$L(\approx)$	AET	EMAE	AET	EMAE
2	40	0.018387	0.0000	350	350	0.113116	0.0060	0.308154	0.0089
	80	0.029383	0.0000	650	650	0.106877	0.0145	0.517740	0.0137
	160	0.063667	0.0000	1350	1350	0.126797	0.0173	0.938203	0.0193
	320	0.188645	0.0000	2720	2720	0.169885	0.0539	1.794071	0.0554
3	40	18.43	0.0000	52500	12100	24.337210	0.0187	98.109933	0.0191
	80	85.21	0.0000	228400	43200	26.354454	0.0250	171.795164	0.0314
	160	386.75	0.0000	923300	191500	30.703085	0.0601	320.648901	0.0822
	320	2064.45	0.0000	3622700	731900	40.323614	0.1110	618.514115	0.1500

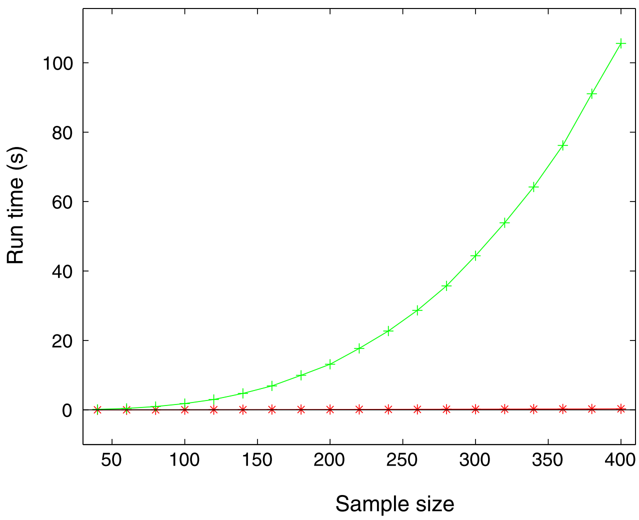


Fig. 4 Speeds comparison between the fast algorithm and Zuo and Lai (2011), where the line '−+' denotes the average execution time of Zuo and Lai (2011), and the line '−∗' denotes that of the proposed algorithm based on 1000 repeated computations. The simulated data are taken from the bivariate standard normal distribution

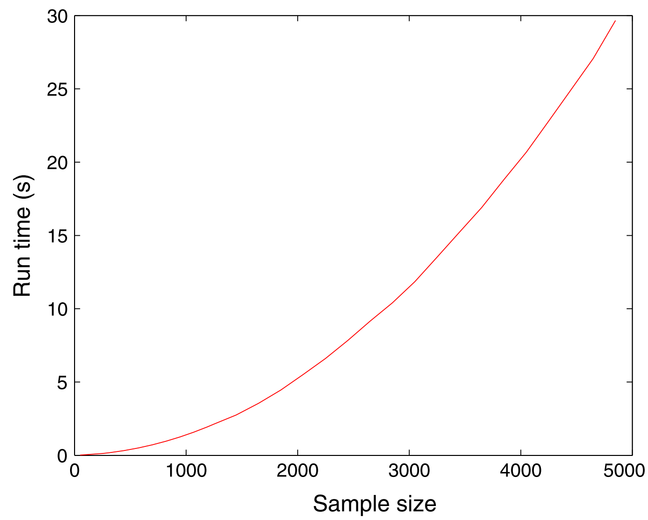


Fig. 5 The average execution time of the proposed algorithm. The simulated data are taken from the bivariate standard normal distribution

repeated computations. Here $n = 40, 60, \dots, 5000$, respectively.

However, it is noteworthy that, when p or n increases, the time requirements are still considerable. Therefore, heuristic techniques are still being expected. However, in any case, the exact algorithm can serve as a benchmark procedure to be compared with any other procedures, in terms of both precision and speed.

Acknowledgements This work was done during Xiaohui Liu's visit to the Department of Statistics and Probability at Michigan State University as a joint PhD student. He thanks his co-advisor Professor Yijun Zuo for stimulating discussions and insightful comments and suggestions and the department for providing excellent studying and working condition. The authors would like to thank two anonymous referees, an associate editor and the editor for their careful reading of the first version of this paper. Their constructive comments led to substantial improvements to the manuscript.

References

Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**, 469–483 (1996)

Bremner, D., Fukuda, K., Marzetta, A.: Primal-dual methods for vertex and facet enumeration. Discrete Comput. Geom. **20**, 333–357 (1998)

Donoho, D.L., Gasko, M.: Breakdown properties of location estimates based on halfspace depth and projected outlyingness. Ann. Stat. **20**, 1808–1827 (1992)

Floyd, R.W., Rivest, R.L.: Algorithm 489: Select. Commun. ACM **18**, 173 (1975)

Halin, M., Paindaveine, D., Šiman, M.: Multivariate quantiles and multiple-output regression quantiles: From L1 optimization to halfspace depth. Ann. Stat. **38**, 635–669 (2010)

Hawkins, D.M., Bradu, D., Kass, G.V.: Location of several outliers in multiple regression data using elemental sets. Technometrics **26**, 197–208 (1984)

Hubert, M., Van der Veeken, S.: Robust classification for skewed data. Adv. Data Anal. Classif. **4**, 239–254 (2010)

Liu, R.Y.: On a notion of data depth based on random simplices. Ann. Stat. **18**, 191–219 (1990)

Liu, R.Y.: Data depth and multivariate rank tests. In: Dodge, Y. (ed.) L1-Statistical Analysis and Related Methods, pp. 279–294. North-Holland, Amsterdam (1992)
Liu, X.H., Zuo, Y.J., Wang, Z.Z.: Exactly computing bivariate projection depth median and contours. Preprint (2011)
Mosler, K., Lange, T., Bazovkin, P.: Computing zonoid trimmed regions of dimension $d > 2$. Comput. Stat. Data Anal. **53**, 2500–2510 (2009)
Paindaveine, D., Šiman, M.: On directional multiple-output quantile regression. J. Multivar. Anal. **102**, 193–392 (2011)
Paindaveine, D., Šiman, M.: Computing multiple-output regression quantile regions. Comput. Stat. Data Anal. **56**, 840–853 (2012a)
Paindaveine, D., Šiman, M.: Computing multiple-output regression quantile regions from projection quantiles. Comput. Stat. **27**, 29–49 (2012b)
Rousseeuw, P.J., Hubert, M.: Regression depth (with discussion). J. Am. Stat. Assoc. **94**, 388–433 (1999)
Rousseeuw, P.J., Leroy, A.: Robust Regression and Outlier Detection, p. 99. Wiley, New York (1987)
Ruts, I., Rousseeuw, P.J.: Computing depth contours of bivariate point clouds. Comput. Stat. Data Anal. **23**, 153–168 (1996)
Serfling, R.: Depth functions in nonparametric multivariate inference. In: Liu, R.Y., Serfling, R., Souvaine, D.L. (eds.) Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 72, pp. 1–16. American Mathematical Society, Providence (2006)
Stahel, W.A.: Breakdown of covariance estimators. Research Report 31, Fachgruppe für Statistik, ETH, Zürich (1981)
Swarup, K.: Linear fractional functionals programming. Oper. Res. **13**, 1029–1036 (1965)
Tukey, J.W.: Mathematics and the picturing of data. In: Proceedings of the International Congress of Mathematicians, pp. 523–531. Cana. Math. Congress, Montreal (1975)
Zuo, Y.J.: Projection based depth functions and associated medians. Ann. Stat. **31**, 1460–1490 (2003)
Zuo, Y.J.: Multidimensional trimming based on projection depth. Ann. Stat. **34**, 2211–2251 (2006)
Zuo, Y.J., Cui, H.J.: Depth weighted scatter estimators. Ann. Stat. **33**, 381–413 (2005)
Zuo, Y.J., Lai, S.Y.: Exact computation of bivariate projection depth and the Stahel-Donoho estimator. Comput. Stat. Data Anal. **55**, 1173–1179 (2011)
Zuo, Y.J., Serfling, R.: General notions of statistical depth function. Ann. Stat. **28**, 461–482 (2000)
Zuo, Y.J., Cui, H.J., He, X.M.: On the Stahel-Donoho estimators and depth-weighted means for multivariate data. Ann. Stat. **32**, 189–218 (2004)