**Metrika manuscript No.**
(will be inserted by the editor)

# Computation of the multivariate Oja median

**T. Ronkainen, H. Oja, P. Orponen**

University of Jyväskylä, Department of Mathematics and Statistics, Finland

Received: date / Revised version: date

**Abstract** The multivariate Oja (1983) median is an affine equivariant multivariate location estimate with high efficiency. This estimate has a bounded influence function but zero breakdown. The computation of the estimate appears to be highly intensive. We consider different, exact and stochastic, algorithms for the calculation of the value of the estimate. In the stochastic algorithms, the gradient of the objective function, the rank function, is estimated by sampling observation hyperplanes. The estimated rank function with its estimated accuracy then yields a confidence region for the true Oja sample median, and the confidence region shrinks to the sample median with the increasing number of the sampled hyperplanes. Regular grids and and the grid given by the data points are used in the construction. Computation times of different algorithms are discussed and compared. For a $k$-variate data set with $n$ observations our exact and stochastic algorithm have rough complexity estimates of $O(kn^k \log n)$ and $O(5^k(1/\varepsilon)^2)$, respectively, where $\varepsilon$ is the radius of confidence $L_\infty$-ball.

**Key words** Multivariate median – multivariate rank – stochastic approximation

## 1 Introduction

Let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be a data set of $n$ $k$-variate observation vectors. The *observation hyperplane* going through the $k$ observations $i_1 < \cdots < i_k$ is given by

$$\left\{ \mathbf{x} \in \mathbb{R}^k \;:\; \det \begin{pmatrix} 1 & \cdots & 1 & 1 \\ \mathbf{x}_{i_1} & \cdots & \mathbf{x}_{i_k} & \mathbf{x} \end{pmatrix} = 0 \right\}.$$

To simplify the notations, let the index $I = (i_1, \dots, i_k)$, $1 \le i_1 < i_2 < \cdots < i_k \le n$ refer to a $k$-subset of observations with indices listed in $I$ and $H(I)$ the hyperplane determined by them. Define then $d_0(I)$ and $\mathbf{d}(I)$ implicitly by

$$\det \begin{pmatrix} 1 & \cdots & 1 & 1 \\ \mathbf{x}_{i_1} & \cdots & \mathbf{x}_{i_k} & \mathbf{x} \end{pmatrix} = d_0(I) + \mathbf{d}^T(I)\mathbf{x},$$

where

$$\mathbf{d}(I) = (d_1(I), \dots, d_k(I))^T$$

and $d_0(I), d_1(I), \dots, d_k(I)$ are the cofactors of the elements in the last column of the matrix. Note also also that the intersection of $k-1$ hyperplanes $H(I_1), \dots, H(I_{k-1})$ is usually (not always) a univariate *observation line*. Finally, the intersection of $k$ hyperplanes $H(I_1), \dots, H(I_k)$ usually (not always) yields a *crossing point*.

The volume of the $k$-variate *simplex* determined by the $k$-set $I$ along with $\mathbf{x}$ is

$$V_I(\mathbf{x}) = \frac{1}{k!} \operatorname{abs}\left\{d_0(I) + \mathbf{d}^T(I)\mathbf{x}\right\}.$$

The *multivariate Oja median* [5] is the vector $\mathbf{T} = \mathbf{T}(X)$ that minimizes the objective function

$$D(\mathbf{x}) = \operatorname*{ave}_I\{V_I(\mathbf{x})\}, \tag{1}$$

the average of the volumes of simplices determined by the candidate $\mathbf{x}$ and all possible $k$-sets of observations $I$. The centered rank function (which is related to this generalization of the multivariate median) is

$$\mathbf{R}(\mathbf{x}) = \nabla D(\mathbf{x}) = \frac{1}{k!} \operatorname*{ave}_I\{S_I(\mathbf{x})\mathbf{d}(I)\} \tag{2}$$

where

$$S_I(\mathbf{x}) = \operatorname{sign}\{d_0(I) + \mathbf{d}^T(I)\mathbf{x}\}$$

is $\pm 1$ and indicates on which side of hyperplane $I$ the point $\mathbf{x}$ is located. Another presentation for the objective function is $D(\mathbf{x}) = \operatorname{ave}_I\{S_I(\mathbf{x})d_0(I)\} + \left(\operatorname{ave}_I\{S_I(\mathbf{x})\mathbf{d}(I)\}\right)^T \mathbf{x}$. The affine equivariant centered rank vectors

$$\mathbf{R}_i = \mathbf{R}(\mathbf{x}_i), \quad i = 1, \dots, n.$$

can be used in classical multivariate analysis problems, namely, in multivariate multisample location (MANOVA) problems, in principal component analysis (PCA), in multivariate multiple regression, in canonical correlation analysis, and in testing for multinormality (See Visuri et al. [8]).

In this paper we wish to find efficient (exact and stochastic) algorithms for the calculation of the value of the sample Oja median $\mathbf{T} = \mathbf{T}(X)$. Our plan is as follows. First, in Section 2, an algorithm for the calculation of the exact value of $\mathbf{T}$ is outlined and discussed. In Section 3, stochastic estimates are considered; two different estimates $\hat{\mathbf{T}}$ of the true value $\mathbf{T}$, based on a random sample from the set of hyperplanes, are constructed. The computation times of different algorithms are compared. The paper is closed with some comments in Section 4. Detailed descriptions of the algorithms are postponed to the Appendix.

## 2 Exact computation

We first describe a deterministic algorithm for computing the exact value of the Oja median. This method is a generalization of the Oja bivariate median algorithm proposed by Niinimaa and Oja [4]. It is known that the median is located among the intersection points of the observation hyperplanes [2]. The bivariate version iteratively optimizes the objective function along observation lines. At each line, the minimizer is the intersection point of the search line and some other observation line. Then the decision is made on whether to stop (a global minimum) or to continue along the crossing line. In most cases there is only one crossing line; however if the crossing point is an original data point then there are $n - 2$ alternative lines left which to follow, and the next one is chosen among those according to some criterion.

There are several obstacles to a straightforward extension of the bivariate method. As the dimension increases, numerical problems make navigation among the hyperplane intersections less reliable. This leads us to make some changes in the algorithm design. In order to overcome the numerical difficulties, our algorithm is designed to use double bookkeeping for geometrical objects, using both numerical and combinatorial presentations of the objects involved. In the combinatorial presentation, each object carries with it a construction recipe in the form of an integer index set. For example, if the hyperplane in $\mathbb{R}^3$ spanned by observations $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ intersects the line through $\mathbf{x}_4$ and $\mathbf{x}_5$, the intersection point then has index presentation $\{\{1, 2, 3\}, \{4, 5\}\}$. This kind of design is needed to count and construct observation lines. However, details of this kind of combinatorial geometry are not the focus of the paper and thus are skipped.

The exact median algorithm (see Appendix A.1) starts by choosing an observation point close to the centre (the mean vector) of the data cloud. Then a set of $k - 1$ hyperplanes going through the chosen point are picked up randomly and their intersection is ensured to be a line. The most time consuming part of the algorithm is to minimize the objective function (1) on the line. Our approach to this problem will be described later. When the minimum point is found, the next task is to choose another line and repeat the process until no improvement is achieved. The point with the minimum value is always located at the intersection of the line and an observation hyperplane.

In the bivariate case there are only two different types of crossing points. The point is either an intersection of two or $n - 1$ observation lines. In the latter case the crossing point is an original observation. Three-dimensional data adds one more possibility: for instance a point with index presentation $\{\{i_1, i_2\}, \{j_1, j_2, j_3\}\}$ is contained in $n + 1$ different observation lines, namely $\{\{i_1, i_2\}\}$ and $\{\{i_1, i_2, m\}, \{j_1, j_2, j_3\}\}$, where $m \neq i_1$ and $m \neq i_2$. Two other cases are of course an original data point, and an intersection of 3 hyperplanes without common data points as in the 2-dimensional case. Each additional dimension introduces one ore more such new cases, each having a different number of lines to explore.

At the each turning point we count the number of crossing line alternatives. The algorithm described in Appendix stops if the number of lines exceeds our predefined limit. This is not absolutely necessary, because we can instead take a random sample from the collection of possible lines, and not give up unless this sample does not have a line with a lower objective function value. After the construction of the crossing lines we discard all lines already followed. The gradient (2) of the objective function may be used to find the next line with the direction of steepest descent, or with the smallest angle with the gradient. Theoretically it is enough to check this line only; in practice, however, the sum of $\binom{n}{k}$ terms involved in the definition of the gradient is very sensitive to rounding errors. The cumulation of these small errors may cause the gradient vector to point in a slightly wrong direction. The careful stopping criterion of the algorithm reflects this difficulty. We decided to increase the stability of the algorithm by searching along all intersecting lines before trusting the answer.

Also the objective function (1) minimization along a line turns out to be surprisingly difficult. In high dimensions, the observation hyperplanes are typically very dense and split the observation lines to extremely short segments. Although the objective function is convex and thus theoretically easy to minimize on a line, numerically the function contains a small amount of additive noise due to rounding errors. These errors arise from a vast number of infinitesimally small changes (but sometimes erroneously rounded to wrong direction due to lack of precision) at the end of each segment. We failed to apply traditional optimization methods, and use instead the following somewhat slower but numerically more robust algorithm (see Appendix A.2).

Let $E$ denote a $k$-variate box containing all data points. A good choice for $E$ is a slightly enlarged box limited by the componentwise minima and maxima of the data values. The enlargement is usually not important unless the number of data points is quite small. In that case rounding errors may drop some critical points outside the box when performing comparisons with exact limits.

We begin by computing all crossing points of the search line $L$ and the observation hyperplanes. If the crossing point of the line and hyperplane $H$ is not contained in $E$, it can not be the minimum. When calculating the value of the rank function in $E$, the sum over terms corresponding to these type of hyperplanes is constant and have to be computed only once. All other hyperplanes are reordered so that two consecutive hyperplane crossing points in $L$ do not have a third point between them.

The task is now to scan through all crossing points in $E$ and choose the point with the lowest objective function value. We set the first crossing point as a median candidate. Evaluation of the objective function is done by summing together one term for each hyperplane and the constant part which was calculated during hyperplane preprocessing. After that every other intersection point in $E$ is handled in the previously given order and the lowest objective function value is recorded. However, evaluation of the objective function is much easier now, since it is necessary only to update the sign of the term corresponding to the hyperplane which intersects $L$ at the current point.

The space requirement of the algorithm is $O(n^k)$ which is needed to store the representations of $\binom{n}{k}$ hyperplanes. Its time complexity cannot be determined so easily. Each minimization along a line has a complexity of $O(n^k \log n^k) =$

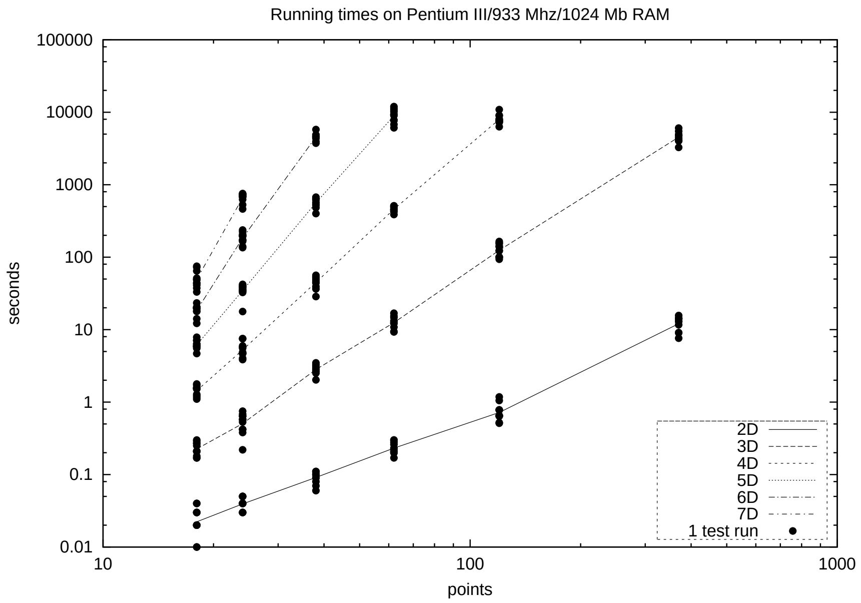

**Fig. 1** Performance of the exact algorithm A.1

$O(kn^k \log n)$, due to the hyperplane sorting involved. This is repeated at least $k$ times, which yields a lower bound of complexity $\Omega(k^2 n^k \log n)$. Simulation experiments suggest that usually $O(k)$ iterations are enough., but the worst case behavior of the algorithm may depend on the distribution of data in a complicated way. (The algorithm and its analysis are similar to the simplex method for linear programming, which is known to be exponential in the worst case, but linear-time on the average under some distributional assumptions [1, pp. 124–128].) Figure 1 presents running times for several multivariate i.i.d. normally distributed test cases.

## 3 Stochastic approximation

As can be seen from the computation time diagrams in Figure 1, the exact algorithm becomes rapidly infeasible as the number of observations $n$, and in particular the dimensionality $k$ of the observation vectors grow. For instance, within a time limit of 1000 seconds (∼28 minutes), our implementation of the exact algorithm was able to process 2-variate data sets containing more than 4000 observation vectors, but for 4-variate data the maximum size of a data set had dropped to about 70 points.

Considerably more efficient computations are possible if the exactness requirement of the algorithm is relaxed somewhat, and a stochastic approximation approach is adopted. The basic idea then is to *sample the collection of observation hyperplanes*, and use these samples to direct the construction of increasingly precise estimates for the location of the Oja median. We present two algorithms based on this general idea: in the first method the median estimates are located at the nodes of a regular, increasingly dense *grid*, whereas in the second method the search for increasingly good median estimates is supported on the initially given observation vectors.

### 3.1 Confidence regions for sample Oja median

Assume that we have a random sample of $n_h$ hyperplanes with indices $I_1, \dots , I_{n_h}$ and we wish to test the null hypothesis $H_0 : \mathbf{T} = \mathbf{z}$, that is, $\mathbf{z}$ is the true sample Oja median. A natural test statistic can then be based on the estimated sample rank function at $\mathbf{z}$,

$$\hat{\mathbf{R}}(\mathbf{z}) = \operatorname*{ave}_{j} \left\{ S_{I_j}(\mathbf{z}) \mathbf{d}(I_j) \right\} .$$

The limiting null distribution of $\sqrt{n_h}\,\hat{\mathbf{R}}(\mathbf{z})$ is a $k$-variate normal distribution with zero mean vector and covariance matrix $C = \text{ave}_I\{\mathbf{d}(I)\mathbf{d}^T(I)\}$ where the sum is over all possible $\binom{n}{k}$ hyperplane indices $I$. A natural estimate of $C$ is $\hat{C} = \text{ave}_j\{\mathbf{d}(I_j)\mathbf{d}^T(I_j)\}$ which is consistent under the null hypothesis. An approximate $100(1-\alpha)$ percent confidence region for $\mathbf{T}$ is then given by

$$\left\{ \mathbf{z} \,:\, n_h \hat{\mathbf{R}}^T(\mathbf{z}) \hat{C}^{-1} \hat{\mathbf{R}}(\mathbf{z}) \leq \chi^2_\alpha(k) \right\} . \tag{3}$$

If $n_h \to \infty$, this confidence region shrinks to a unique point, the Oja median. The problem is, however, the explicit calculation of the confidence region. For approximations, values of the test statistic

$$U(\mathbf{z}) = n_h \hat{\mathbf{R}}^T(\mathbf{z}) \hat{C}^{-1} \hat{\mathbf{R}}(\mathbf{z})$$

are calculated at the nodes of an increasingly dense grid.

### 3.2 Approximation supported on a regular grid

Given a set of $k$-variate observation vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$, our grid-based hyperplane sampling algorithm (Algorithm A.3 in the appendix) is initialized with a regular $k$-dimensional grid $\mathcal{G}_h$, with $5^k$ gridpoints spaced uniformly at distance $h$. The grid spacing $h$ is determined as 1/4 of the $L_\infty$ diameter of the data set, and the grid is centered so that all observation vectors are contained within its extrema.

We then choose some confidence level $1-\alpha$, say $0.95$, sample an initial set of $n_h = n_s$ observation hyperplanes, and compute the values of the test statistic $U(\mathbf{z})$ at each gridpoint $\mathbf{z} \in \mathcal{G}_h$. With an ideal choice of $n_s$ and the sampled hyperplanes, only a single gridpoint $\mathbf{z}_0$ would then remain in the $100(1-\alpha)$ percent confidence region (3) of $T$. Usually this is not the case, but nevertheless we can eliminate a number of gridpoints from $\mathcal{G}_h$ as falling outside the confidence region. This results in a reduced grid $\mathcal{G}'_h = \mathcal{G}_h$. (The choice of a good value for the sample batch size parameter $n_s$ depends on the characteristics of the data. We have simply picked a value $n_s = 20$ for our simulations. However the average running time of the algorithm seems to be quite insensitive to the choice of $n_s$, as long as it is neither very small nor big.)

We now add another set of $n_s$ hyperplanes to our sample (consequently $n'_h = n_h + n_s$), update the values of the test statistic $U(\mathbf{z})$ at the remaining gridpoints $\mathbf{z} \in \mathcal{G}'_h$ (Algorithm A.4), eliminate the gridpoints falling outside the decreased confidence region (3) etc. This process is iterated until only one point $\mathbf{z}_0$ remains active in the grid. If some added sample of $n_s$ hyperplanes causes the number of active gridpoints to go to zero, then this sample is withdrawn and a new sample is generated instead.

When only one gridpoint $\mathbf{z}_0$ remains, the grid is refined by halving the spacing from $h$ to $h/2$ in the neighborhood of $\mathbf{z}_0$, and omitting that part of the old grid that is at $L_\infty$ distance greater than $h$ from $\mathbf{z}_0$. The new $5^k$-node grid $\mathcal{G}_{h/2}$ is centered at $\mathbf{z}_0$, and initial test function estimates at those nodes of $\mathcal{G}_{h/2}$ that were not contained in $\mathcal{G}_h$ are obtained by linear interpolation (see Algorithm A.3). The hyperplane sampling and grid reduction process described above is then resumed with the grid $\mathcal{G}_{h/2}$. When grid $\mathcal{G}_{h/2}$ is reduced to a single point, it is again refined in the neighborhood of this point to a grid $\mathcal{G}_{h/4}$ etc. The procedure is repeated until the grid spacing becomes denser than some predefined parameter $\varepsilon > 0$. At this point we know the location of the Oja median of our data set with $L_\infty$ precision $\varepsilon$, to within confidence $1 - \alpha$.

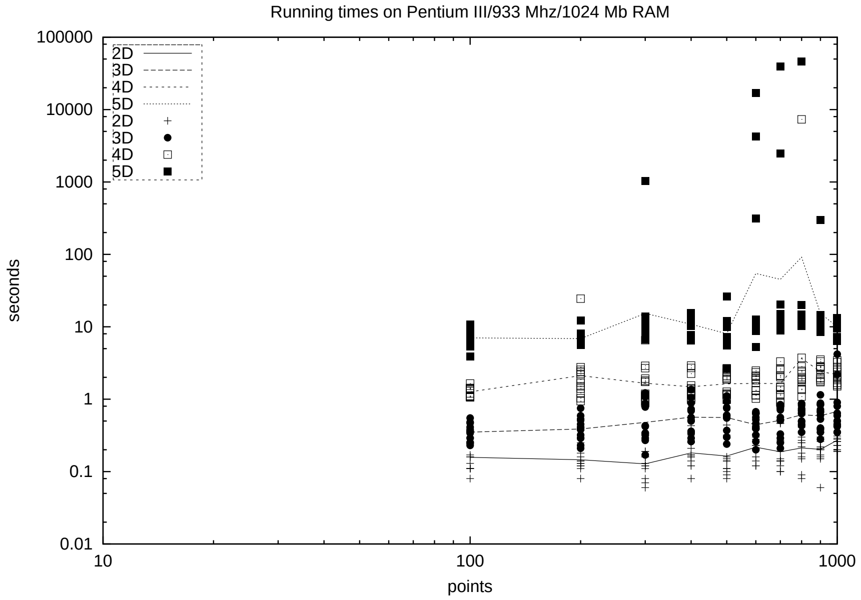

**Fig. 2** Performance of the grid-based approximate algorithm A.3

As can be seen from Figure 2, the computation time of this stochastic approximate algorithm is practically independent of the number $n$ of the observation vectors. The only factors affecting the computation time, in addition to the confidence parameter $\alpha$, are the dimensionality $k$ of the data and the precision $\varepsilon$ required in the stopping criterion.

A rough estimate of the time complexity of the method can be obtained by noting that if $n_h$ hyperplane samples are needed to achieve a confidence region with $L_\infty$ radius less than $h$, i.e. to reduce a grid $\mathcal{G}_h$ to a single point, then approximately $4n_h$ samples are needed to reduce the radius further to below $h/2$. In other words, between any two grid refinements the number of requisite hyperplane samples grows roughly by a factor of four.

The grid spacing decreases below $\varepsilon$ after $O(\log_2 1/\varepsilon)$ sampling stages. At each stage $i \geq 0$, the number of hyperplanes involved in the calculations is $O(4^i n_h)$, where $n_h$ is the number of hyperplanes required for the first refinement. The number of active gridpoints at which the test function needs to be evaluated equals $5^k$ at the beginning of each stage. Altogether this yields an estimate of $O(5^k 4^{\log_2 1/\varepsilon} n_h) = O(5^k (1/\varepsilon)^2)$ for the total asymptotic complexity of the algorithm. Figure 2 provides experimental support for this estimate. (The parameter values used in the simulations were $\alpha = 0.05$, $\varepsilon = 0.1$.)

### *3.3 Approximation supported on data points*

An undesirable consequence of supporting the median algorithm on a rigid grid structure, as above, is the induced exponential dependence of the algorithm's running time on the dimensionality of the problem. An alternative idea is then to use in place of the grid the observation vectors themselves. This approach replaces the algorithm's exponential dependence on dimensionality by a simple linear dependence on the number of data points, and results in huge reductions in computation time for high-dimensional data. However, the precision of the final estimate is now determined by the density of the given set of observation vectors. We now describe a computation method based on this idea (Algorithm A.5 in the appendix).

The first phase of Algorithm A.5 proceeds exactly as in the grid-based Algorithm A.3, except that instead of the grid $\mathcal{G}_h$, the support set $Z$ used to obtain the approximation initially consists of the given observation vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Also, rather than refining the support set $Z$ to a predetermined precision, Algorithm A.5 aims to thin out $Z$ by hyperplane sampling until a set of exactly $k+1$ vectors remain in the confidence region (3). The $k$-simplex spanned by these $k+1$ highest-confidence observation vectors then provides an estimate for the location of the median. (If desired, a point estimate can be obtained by linear interpolation of the rank function estimate over the simplex.) As in the univariate case, the precision of the estimate is thus determined by the density of the data cloud around the median.

Thus, batches of $n_s$ hyperplanes are sampled and the support set $Z$ reduced as in Algorithm A.3 until its size decreases below some threshold value $|Z| \leq n_2$. Then the algorithm switches to sampling a single hyperplane at a time and reducing the remaining set $Z$ with that until a size of $|Z| = k+1$ is reached.

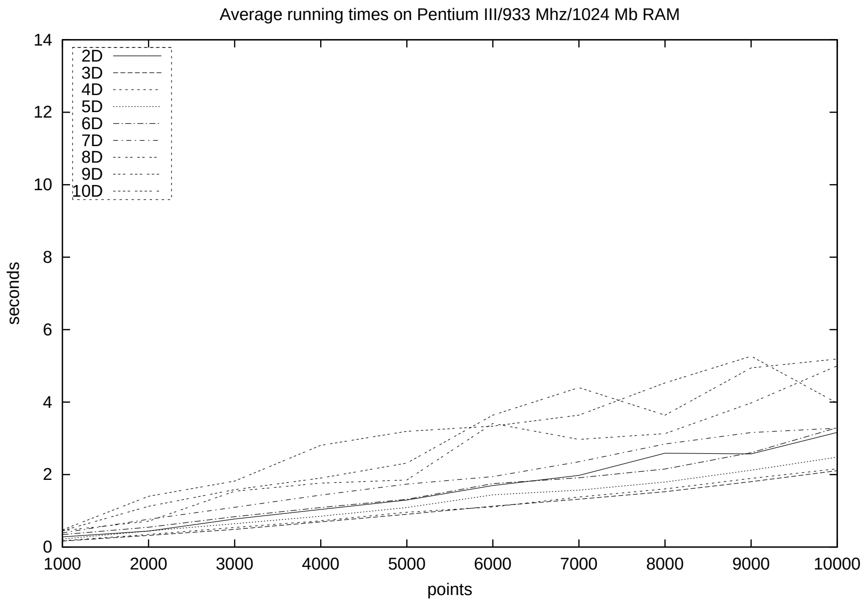

**Fig. 3** Performance of the data-point based approximate algorithm A.5

As can be seen from Figure 3.3, this algorithm is overwhelmingly faster than the grid-based method for high-dimensional data. For instance even 10-variate data sets with 10000 observation vectors were handled faster than sets of 5-variate data using the grid-based method. A limitation induced by using the observation vectors as gridpoints, on the other hand, is that while Algorithm A.3 yields a reliable estimate of the location of the median up to a given precision, in some pathological cases the true median may actually be located arbitrarily far away from the $k$-simplex provided by Algorithm A.5. (Consider, for instance data distributed uniformly on a spherical surface of dimension $k-1$.)

## 4 Final comments

We introduced different (exact and stochastic) algorithms for the calculation of the value of the sample Oja median $\mathbf{T}$. The problem then naturally is, what are the losses (in bias and efficiency) and the gains (in computing time) when using stochastic estimates? In the paper we considered only the computation times of the algorithms. Note that stochastic estimate $\hat{\mathbf{T}}$ is an estimate of the true population value $\boldsymbol{\mu}$ also and in the comparisons the bias

$$E(\hat{\mathbf{T}}-\boldsymbol{\mu}) = E[E(\hat{\mathbf{T}}-\mathbf{T}|\mathbf{T})] - E(\mathbf{T}-\boldsymbol{\mu})$$

as well as the covariance matrix

$$\mathrm{Cov}(\hat{\mathbf{T}}) = \mathrm{Cov}[E(\hat{\mathbf{T}}|\mathbf{T})] + E[\mathrm{Cov}(\hat{\mathbf{T}}|\mathbf{T})]$$

should be considered. As 'an estimate of an estimate is an estimate', the two different stochastic versions should be compared in simple simulation studies.

In practice, also an estimate of the accuracy (covariance matrix) of the true Oja median is called for. For this purpose, it is possible to estimate simultaneously the value of the Oja median and the values of the bootstrap sample medians. These bootstrap sample medians may then be used in the usual way to construct the covariance matrix estimate. See also Nadar et al. [3] for the estimation of the limiting covariance matrix.

The computing time of the stochastic algorithm appears to be sensitive to the choice of the hyperplane sample batch size $n_s$ (see the strange computing times in Figure 2). It is possible that sometimes the values of $n_{h0}$, $\hat{C}_0$ and $\hat{\mathbf{R}}_0(\mathbf{z})$ are saved just before the number of active gridpoints is reduced to one. Then with high probability no active gridpoints are left after a new batch. The problem may be circumvented by a runtime adjustment of the batch size $n_s$. Some additional robustness would be achieved if, in the sampling process, sampling probabilities depending on the 'size' of the hyperplane were utilized. The rank function estimate is then the regular Horwitz-Thompson estimate with the easily calculable covariance matrix estimate.

We have also experimented with a parallel version of the exact algorithm.We used simultaneous processes to compute several minimization tasks A.2 at the same time. One process is needed to coordinate progress by distributing tasks to other processes. This is an almost idle job when compared to the heavy calculations and can be executed without noticeable performance loss in one of the processors that participates in the computation. Speedup is good up to $k$ processors but after that additional processors do not bring much benefit. This is a consequence of the fact that at each step there are usually only $k$ observation line through the current crossing point and extra processors can only explore older lines hoping to find some random shortcut.

The approximate algorithms can also gain benefit from distributed computing, but we have not yet implemented it. Algorithm A.4 is the core of the both algorithms. Because update formulas for rank and covariance estimates are additive, each of the $P$ processes can sample its own set of $\lceil n_s/P \rceil$ hyperplanes. After collecting together changes in $\hat{\mathbf{R}}$, $\hat{C}$ and $n_h$, grid nodes can be divided evenly among the processes, each updating about $|Z|/P$ test values $U(\mathbf{z}_i)$.

## A Appendix: Description of algorithms

### *A.1 Compute the exact Oja median.*

```
Input: Data set $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ in $\mathbb{R}^k$.
Input: Maximum number of observation lines to scan at each step $max_{n_L} \geq k$
Output: Exact Oja median $\mathbf{T}$ or give up with probability going to zero as $n$ increases.
1: Precalculate observation hyperplanes $H(I)$, where
       $$I = (1, 2, \dots, k), \dots, (n-k+1, \dots, n-1, n).$$
```

```
 2: Choose random observation $\mathbf{x}_j$ near $\bar{x}$.
 3: Select random indices $I_1, \dots, I_{k-1}$ each containing $j$.
 4: Set $L \leftarrow H(I_1) \cap \dots \cap H(I_{k-1})$.
 5: if $\dim(L) > 1$ then
 6:     Goto 3.
 7: end if
 8: Apply algorithm A.2 to compute $\hat{\mathbf{T}} \leftarrow \arg\min_{\mathbf{t} \in L} D(\mathbf{t})$.
 9: Set the median candidate $\mathbf{T} \leftarrow \hat{\mathbf{T}}$.
10: Initialize the collection of investigated lines $\mathcal{L} \leftarrow \{L\}$.
11: Let $n_L$ be the number of the observation lines containing $\hat{\mathbf{T}}$.
12: if $n_L > max_{n_L}$ then
13:     Give up. There are too many possibilities.
14: end if
15: Construct these observation lines $\mathcal{L}' \leftarrow L_1, \dots, L_{n_L}$.
16: Set $\mathcal{L}' \leftarrow \mathcal{L}' \cap \mathcal{L}$.
17: while $\mathcal{L}' \neq \emptyset$ do
18:     Find the line of deepest descent
            $L \leftarrow \arg\max_{L \in \mathcal{L}'} \left\{ \left| \mathbf{R}^T(\hat{\mathbf{T}}) \mathbf{u}_L \right| \right\}$,
        where $\mathbf{u}_L$ is the unit vector in the direction of $L$.
19:     Apply algorithm A.2 to have $\hat{\mathbf{T}} \leftarrow \arg\min_{\mathbf{t} \in L} D(\mathbf{t})$.
20:     Update $\mathcal{L} \leftarrow \mathcal{L} \cup \{L\}$ and $\mathcal{L}' \leftarrow \mathcal{L}' - \{L\}$.
21:     if $D(\hat{\mathbf{T}}) < D(\mathbf{T})$ then
22:         Goto 9.
23:     end if
24: end while
25: Return $\mathbf{T}$.
```

### *A.2 Minimize $D$ on the observation line.*

```
Input:  Precalculated observation hyperplanes $H(I)$.
Input:  Observation line $L$.
Input:  Enclosing box $E = [\mathbf{x}_{min}, \mathbf{x}_{max}]$.
Output: The minimum $\mathbf{t} = \arg\min_{\mathbf{t} \in L} D(\mathbf{t})$.
 1: Choose any point $\mathbf{t}_0 \in E \cap L$.
 2: Initialize $D_0 \leftarrow 0$, $\mathbf{D} \leftarrow \mathbf{0}$ and $\mathcal{H} \leftarrow \emptyset$.
 3: for all $I$ do
 4:     if $H(I) \cap L \subset E$ then
 5:         Add $\mathcal{H} \leftarrow \mathcal{H} \cup \{I\}$.
 6:     else
 7:         $\mathbf{D} \leftarrow \mathbf{D} + \frac{1}{k!} S_I(\mathbf{t}_0) \mathbf{d}(I)$
 8:         $D_0 \leftarrow D_0 + \frac{1}{k!} S_I(\mathbf{t}_0) d_0(I)$.
 9:     end if
10: end for
```

```
Sort hyperplane indices $I \in \mathcal{H}$ according to intersection points, i.e. if $L = \{\mathbf{L}_0 + \beta\mathbf{u}_L : \beta \in \mathbb{R}\}$ and we have $H(I) \cap L = \{\mathbf{L}_0 + \beta_I\mathbf{u}_L\}$ and $H(J) \cap L = \{\mathbf{L}_0 + \beta_J\mathbf{u}_L\}$ for some $\beta_I$ and $\beta_J$, then $I < J \iff \beta_I < \beta_J$. Set $n_P \leftarrow |\mathcal{H}|$ and denote the order by $I_{(1)}, I_{(2)}, \ldots, I_{(n_P)}$.
Set $\{\mathbf{y}_1\} \leftarrow L \cap H_{(1)}$.
Set $\{\mathbf{y}_{n_P}\} \leftarrow L \cap H_{(n_P)}$.
Set the potential minimum $\hat{\mathbf{T}} \leftarrow \mathbf{y}_1$.
Compute $\mathbf{D} \leftarrow \mathbf{D} + \frac{1}{k!}\sum_{I \in \mathcal{H}} S_I(\hat{\mathbf{T}})\mathbf{d}(I)$ and
$D_0 \leftarrow D_0 + \frac{1}{k!}\sum_{I \in \mathcal{H}} S_I(\hat{\mathbf{T}})d_0(I)$.
Evaluate $D(\hat{\mathbf{T}})$, i.e. set $\hat{D} \leftarrow \mathbf{D}^T\hat{\mathbf{T}} + D_0$.
**for all** $i = 2, \ldots, n_P$ **do**
    Set[1]
        $$\mathbf{D} \leftarrow \mathbf{D} - \frac{1}{k!}S_{I_{(i-1)}}(\mathbf{y}_1)\,\mathbf{d}(I_{(i-1)}) + \frac{1}{k!}S_{I_{(i-1)}}(\mathbf{y}_{n_P})\,\mathbf{d}(I_{(i-1)})$$
    Set
        $$D_0 \leftarrow D_0 - \frac{1}{k!}S_{I_{(i-1)}}(\mathbf{y}_1)\,d_0(I_{(i-1)}) + \frac{1}{k!}S_{I_{(i-1)}}(\mathbf{y}_{n_P})\,d_0(I_{(i-1)}).$$
    Set $\{\mathbf{t}\} \leftarrow L \cap H(I_{(i)})$.
    **if** $\mathbf{D}^T\mathbf{t} + D_0 < \hat{D}$ **then**
        Set $\hat{\mathbf{T}} \leftarrow \mathbf{t}$ and $\hat{D} \leftarrow \mathbf{D}^T\mathbf{t} + D_0$.
    **end if**
**end for**
Return $\hat{\mathbf{T}}$.
```

### *A.3 Compute an estimate of the Oja median.*

```
*Input:* Data set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^k$.
*Input:* Number of the sampled hyperplanes between successive grid updatings $n_s \geq 1$.
*Input:* Confidence level $0 < 1 - \alpha < 1$.
*Input:* Radius of the confidence $L_\infty$-ball $\varepsilon > 0$.
*Output:* Estimate $\hat{\mathbf{T}}$ of the Oja median $\mathbf{T}(X)$, s.t. $\mathbb{P}(\|\hat{\mathbf{T}} - \mathbf{T}(X)\|_\infty < \varepsilon) = 1 - \alpha$.
Find the enclosing box of the data
    $$\mathbf{x}_{min} = (\min\{x_{11}, \ldots, x_{n1}\}, \ldots, \min\{x_{1d}, \ldots, x_{nk}\})$$
and
    $$\mathbf{x}_{max} = (\max\{x_{11}, \ldots, x_{n1}\}, \ldots, \max\{x_{1d}, \ldots, x_{nk}\}).$$
Set the grid spacing $h \leftarrow \max\{(x_{max\,i} - x_{min\,i}\})/4 : i = 1, \ldots, k\}$.
Set the center of the grid $\mathbf{z}_0 \leftarrow \frac{1}{2}(\mathbf{x}_{min} + \mathbf{x}_{max})$.
```

[1] This assignment is numerically safer than the shorter, mathematically equivalent formula $\mathbf{D} \leftarrow \mathbf{D} - \frac{2}{k!}S_{I_{(i-1)}}(\mathbf{y}_1)\,\mathbf{d}(I_{(i-1)})$.

```
 4: Select the regular grid
        $Z \leftarrow \prod_{i=1}^{k} \{z_{0i} - 2h, z_{0i} - h, z_{0i}, z_{0i} + h, z_{0i} + 2h\}$ .
 5: Initialize $n_h \leftarrow 0$, $\hat{C} \leftarrow 0$ $k \times k$ matrix and $\forall\, \mathbf{z} \in Z \,:\, \hat{\mathbf{R}}(\mathbf{z}) \leftarrow \mathbf{0}$ $k$-vector.
 6: Save values $n_{h0} \leftarrow n_h, \hat{C}_0 \leftarrow \hat{C}$ and $\forall\, \mathbf{z} \in Z \,:\, \hat{\mathbf{R}}_0(\mathbf{z}) \leftarrow \hat{\mathbf{R}}(\mathbf{z})$.
 7: repeat
 8:     Update rank estimates with algorithm A.4.
 9:     Select $Z \leftarrow \{\mathbf{z} \in Z \,:\, U(\mathbf{z}) \le \chi^2_\alpha(k)\}$.
10: until $|Z| \le 1$.
11: if $|Z| = 0$ then
12:     Restore values $n_h \leftarrow n_{h0}, \hat{C} \leftarrow \hat{C}_0$ and $\forall\, \mathbf{z} \in Z \,:\, \hat{\mathbf{R}}(\mathbf{z}) \leftarrow \hat{\mathbf{R}}_0(\mathbf{z})$.
13:     Goto 7.
14: else if $h > \varepsilon$ then
15:     Refine the grid $h \leftarrow \frac{h}{2}$, $\mathbf{z}_0 \leftarrow \mathbf{z} \in Z$ and
            $Z' \leftarrow \prod_{i=1}^{k} \{z_{0i} - 2h, z_{0i} - h, z_{0i}, z_{0i} + h, z_{0i} + 2h\}$ .
16:     Initialize rank estimates $\forall\, \mathbf{z}' \in Z'$:
            $\hat{\mathbf{R}}(\mathbf{z}') = \begin{cases} \hat{\mathbf{R}}(\mathbf{z}), & \text{if } \mathbf{z}' \in Z, \\ \text{ave}\{\hat{\mathbf{R}}(\mathbf{z}) \,:\, \mathbf{z} \in Z \text{ and } \|\mathbf{z} - \mathbf{z}'\|_\infty = h\}, & \text{otherwise.} \end{cases}$
17:     Set $Z \leftarrow Z'$ and goto 6.
18: end if
19: Return $\mathbf{z} \in Z$.
```

*A.4 Update rank estimates.*

*Input:* Number of the hyperplanes $n_s$ to be sampled.
*Input:* Collection of points $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$, where ranks should be estimated.
*Input:* Current rank estimates $\hat{\mathbf{R}}(\mathbf{z}_1), \ldots, \hat{\mathbf{R}}(\mathbf{z}_m)$, covariance $\hat{C}$ and the number of the already sampled hyperplanes $n_h$.
*Input:* Observations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^k$ (these are necessary for computing $\mathbf{d}(I)$).
*Output:* Updated $\hat{\mathbf{R}}(\mathbf{z}_1), \ldots, \hat{\mathbf{R}}(\mathbf{z}_m)$, $\hat{C}$ and $n_h$.
*Output:* Test statistic values $U(\mathbf{z}_1), \ldots, U(\mathbf{z}_m)$.

```
for $n_s$ times do
    Choose randomly $I = (i_1, \ldots, i_k)$, $i_j < i_{j+1}$, $1 \le i_j \le n$.
    Set $n_h \leftarrow n_h + 1$.
    for all $\mathbf{z} \in Z$ do
        Set $\hat{\mathbf{R}}(\mathbf{z}) \leftarrow \frac{n_h - 1}{n_h} \hat{\mathbf{R}}(\mathbf{z}) + \frac{1}{n_h} S_I(\mathbf{z})\mathbf{d}(I)$.
    end for
    Set $\hat{C} \leftarrow \frac{n_h - 1}{n_h} \hat{C} + \frac{1}{n_h} \mathbf{d}(I)\mathbf{d}^T(I)$.
```

```
 8: end for
 9: for all $\mathbf{z} \in Z$ do
10:     Set $U(\mathbf{z}) \leftarrow n_h \, \hat{\mathbf{R}}(\mathbf{z})^T \hat{C}^{-1} \hat{\mathbf{R}}(\mathbf{z})$.
11: end for
```

### *A.5 Find minimal simplex approximating the Oja median.*

```
Input: Observations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^k$.
Input: Confidence level $0 < 1 - \alpha < 1$.
Input: Number of the sample hyperplanes during the first phase $n_s \geq 1$.
Input: How many gridpoints to pass to the second phase $n_2 \geq k + 1$.
 1: Initialize $n_h \leftarrow 0$, $\hat{C} \leftarrow 0$ $k \times k$ matrix and $\forall\, \mathbf{z} \in Z \,:\, \hat{\mathbf{R}}(\mathbf{z}) \leftarrow \mathbf{0}$ $k$-vector.
 2: (First phase) Set $Z \leftarrow \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.
 3: while $|Z| > n_2$ do
 4:     Use algorithm A.4 to update rank estimates.
 5:     Select $Z \leftarrow \{\mathbf{z} \in Z \,:\, U(\mathbf{z}) \leq \chi^2_\alpha(k)\}$.
 6: end while
 7: If $|Z| < k + 1$, we have too small $n_2$ value. Increase it and try again.
 8: (Second phase) Set $n_s \leftarrow 1$.
 9: if $|\{\mathbf{z} \in Z \,:\, U(\mathbf{z}) \leq \chi^2_\alpha(k)\}| > k + 1$ then
10:     Sample one hyperplane with algorithm A.4.
11:     Goto 9.
12: end if
13: Find $\mathbf{z}_{(1)}, \ldots, \mathbf{z}_{(k+1)}$ with the lowest test statistic values $U(\mathbf{z})$.
14: Denote the simplex $S \leftarrow$ simplex$(\mathbf{z}_{(1)}, \ldots, \mathbf{z}_{(k+1)})$.
15: if int$(S) \cap Z \neq \emptyset$ then
16:     Sample one hyperplane with algorithm A.4.
17:     Goto 9.
18: end if
19: Return $\mathbf{z}_{(1)}, \ldots, \mathbf{z}_{(k+1)}$.
```

Implementation of these algorithms are available at

`http://www.jyu.fi/∼tojuro/`

as an SPlus module written in C++.

## References

1. Bertsimas, D., and Tsitsiklis, J. N. *Introduction to Linear Optimization.* Athena Scientific, Belmont, MA, 1997.
2. Hettmansperger, T.P., Möttönen, J. and Oja, H. (1999a). The geometry of the affine invariant multivariate sign and rank methods. *J. Nonparam. Statist.*, **11**, 271–285.
3. Nadar, M., Hettmansperger, T.P. and Oja, H. (2001). The asymptotic variance of the Oja median. Submitted.

4. Niinimaa, A., Oja, H., and Nyblom, J. (1992). Algorithm AS277: The Oja bivariate median. *J. Royal Statist. Society, Ser. B*, **41**, 611–633.
5. Oja, H. (1983). Descriptive statistics for multivariate distributions. *Stat. and Prob. Letters*, **1**, 327–332.
6. Oja, H. (1999). Affine invariant multivariate sign and rank tests and corresponding estimates: a review. *Scand. J. Statist.*, **26**, 319–343.
7. Ollila, E., Hettmansperger, T.P. and Oja, H. (2001a). Estimates of the regression coefficients based on the sign covariance matrix. Submitted.
8. Visuri, S., Ollila, E., Koivunen, V., Möttönen, J. and Oja, H. (2001). Affine equivariant multivariate rank methods. Conditionally accepted.