A Rejection Technique for Sampling from Log-Concave Multivariate Distributions

Josef Leydold
University of Economics and Business Administration
Department for Applied Statistics and Data Processing
Augasse 2–6, A-1090 Vienna, Austria
email: `Josef.Leydold@statistik.wu-wien.ac.at`

March 1998

Abstract

Different universal methods (also called automatic or black-box methods) have been suggested to sample from univariate log-concave distributions. The description of a suitable universal generator for multivariate distributions in arbitrary dimensions has not been published up to now. The new algorithm is based on the method of transformed density rejection. To construct a hat function for the rejection algorithm the multivariate density is transformed by a proper transformation T into a concave function (in the case of log-concave density $T(x) = \log(x)$.) Then it is possible to construct a dominating function by taking the minimum of several tangent hyperplanes which are transformed back by T^{-1} into the original scale. The domains of different pieces of the hat function are polyhedra in the multivariate case. Although this method can be shown to work, it is too slow and complicated in higher dimensions. In this paper we split the $\mathbb{R}^n$ into simple cones. The hat function is constructed piecewise on each of the cones by tangent hyperplanes. The resulting function is not continuous any more and the rejection constant is bounded from below but the setup and the generation remains quite fast in higher dimensions, e.g. $n = 8$. The paper describes the details how this main idea can be used to construct algorithm TDRMV() that generates random tuples from multivariate log-concave distribution with a computable density. Although the developed algorithm is not a real black box method it is adjustable for a large class of log-concave densities.

CR Categories and Subject Descriptors: G.3 [Probability and Statistics]: Random number generation

General Terms: Algorithms

Additional Key Words and Phrases: Rejection method, multivariate log-concave distributions, universal method

Contents

1 Introduction

For the univariate case there is a large literature on generation methods for standard distributions (see e.g. [Dev86] and [Dag88]) and in the last years some papers appeared on universal (or black-box) methods (see [Dev86, chapter VII], [GW92], [Ahr95], [Hör95a], [HD94] and [ES97]); these are algorithms that can generate random variates from a large family as long as some information (typically the mode and the density of the specific distribution) are available.

For the generation of variates from bivariate and multivariate distributions papers are rare. Well known and discussed are only the generation of the multinormal and of the Wishart distribution (see e.g. [Dev86] and [Dag88]). Several approaches to the problem of generating multivariate random tuples exist, but these have some disadvantages:

- The multivariate extension of the ratio of uniforms methods as in [SV87] or [WGS91]. This method can be reformulated as rejection from a small family of table-mountain shaped multivariate distributions. This point of view is not included in these two papers but it is useful as it clarifies the question why the acceptance probability becomes poor for high correlation. This disadvantage of the method is already mentioned in [WGS91]. The practical problem how to obtain the necessary multivariate rectangle enclosing the region of acceptance for the ratio of uniforms method is not discussed in [SV87] nor in [WGS91] and seems to be difficult for most distributions.

- The conditional distribution method. It requires the knowledge of and the ability to sample from the marginal and the conditional distributions (see [Dev86, chapter XI.1.2]).

- The decomposition and rejection method. A majorizing function (also called hat-function) suggested for the multivariate rejection method is the product of the marginal densities (in [Dag88]). It is not clear at all how to obtain the necessary rejection constant α.

- Development of new classes of multivariate distributions, which are easy to generate. It is only necessary (and possible) to specify the marginal distribution and the degree of dependence measured by some correlation coefficient (see the monograph [Joh87]). This idea seems to be attractive for most simulation practitioners interested in multivariate distributions but it is no help if variates from a distribution with given density should be generated.

- Recently Devroye [Dev97] has developed algorithms for ortho-unimodal densities. But this paper leaves the generation of log-convave distributions as open problem.

- Sweep-plane methods for log-concave (and T-concave) distributions are described recently in [Hör95b] for bivariate case and in [LH98] for the multivariate case. These algorithms use the idea of transformed density rejection which is presented in a first form in [Dev86, chapter VII.2.4] and with a different set-up in [GW92].

To our knowledge these two algorithms are the only universal algorithms in the literature for multivariate distributions with given densities. (In [Dev86, chapter XI.1.3] it is even stressed that no general inequalities for multivariate densities are available, a fact which makes the design of black-box algorithms, that are similar to those developed in [Dev86] for the univariate case, impossible.)

Although the algorithm in [LH98] works, it is very slow, since the domain of the density f is decomposed in polyhedra. This is due to the construction of the hat function, where we take the pointwise minimum of tangent hyperplanes. In this paper we again use transformed density rejection and the sweep-plane technique to derive a much more efficient algorithm. The main idea is to decompose the domain of the density in cones first and then compute tangent hyperplanes in this cones. The resulting hat function is not continuous any more and the rejection constant is bounded from below, but the setup as well as the sampling from the hat function is much faster than in the original algorithm.

Section 2 explains the method and gives all necessary mathematical formulae. Section 3 provides all details of the algorithm. Section 4 discusses how to improve and extend the main idea of the algorithm (e.g. to T-concave distributions, bounded domain) and section 5 reports the computational experience we have had with the new algorithm.

2 The method

2.1 Transformed density rejection

Density. We are given a multivariate distribution with differentiable density function

$$f\colon D \to [0,\infty), \qquad D \subseteq \mathbb{R}^n, \qquad \text{with mode } \mathbf{m}. \tag{1}$$

To simplify the development of our method we assume $D = \mathbb{R}^n$, $\mathbf{m} = 0$ and $f \in \mathcal{C}^1$. In §4 we extend the algorithm so that these requirements can be dropped.

Transformation. To design an universal algorithm utilizing the rejection method it is necessary to find an automatic way to construct a hat function for a given density. Transformed density rejection introduced under a different name in [GW92] and generalized in [Hör95a] is based on the idea that the density f is transformed by a monotone T (e.g. $T(x) = \log(x)$) in such a way that (see [Hör95a]):

(T1) $\tilde{f}(\mathbf{x}) = T(f(\mathbf{x}))$ is concave (we then say "f is *T-concave*");

(T2) $\lim_{x\to 0} T(x) = -\infty$;

(T3) $T(x)$ is differentiable and $T'(x) > 0$, which implies T^{-1} exists; and

(T4) the volume under the hat is finite.

Hat. It is then easy to construct a hat $\tilde{h}(\mathbf{x})$ for $\tilde{f}(\mathbf{x})$ as the minimum of N tangents. Since $\tilde{f}(\mathbf{x})$ is concave we clearly have $\tilde{f}(\mathbf{x}) \leq \tilde{h}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$. Transforming $\tilde{h}(\mathbf{x})$ back into the original scale we get $h(\mathbf{x}) = T^{-1}(\tilde{h}(\mathbf{x}))$ as majorizing function or hat for f, i.e. with $f(\mathbf{x}) \leq h(\mathbf{x})$. Figure 1 illustrates the situation for the univariate case by means of the normal distribution and the transformation $T(x) = \log(x)$. The left hand side shows the transformed

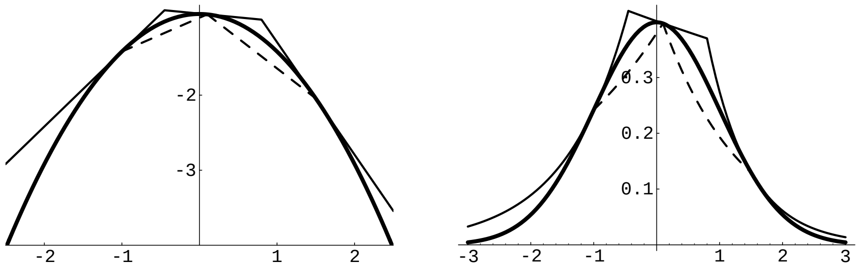

Figure 1: hat function for univariate normal density

density with three tangents. The right hand side shows the density function with the resulting hat. (The dashed lines are simple lower bounds for the density called squeezes in random variate generation. Their use reduces the number of evaluations of f. Especially if the number of touching points is large and the evaluation of f is slow the acceleration gained by the squeezes can be enormous.)

Rejection. The basic form of the multivariate rejection method is given by algorithm Rejection().

Algorithm 1 Rejection()

1: **Set-up:** Construct a hat-function $h(\mathbf{x})$.
2: Generate a random tuple $\mathbf{X} = (X_1, \ldots, X_n)$ with density proportional to $h(\mathbf{X})$ and a uniform random number U.
3: If $U h(\mathbf{X}) \leq f(\mathbf{X})$ return $\mathbf{X}$ else go to 2.

The main idea of this paper is to extend transformed density rejection as described in [Hör95a] to the multivariate case.

2.2 Construction of a hat function

Tangents. Let $\mathbf{p}_i$ be points in $D \subseteq \mathbb{R}^n$. In the multivariate case the tangents of the transformed density $\tilde{f}(\mathbf{x})$ at $\mathbf{p}_i$ are the hyperplanes given by

$$\ell_i(\mathbf{x}) = \tilde{f}(\mathbf{p}_i) + \langle \nabla \tilde{f}(\mathbf{p}_i), (\mathbf{x} - \mathbf{p}_i) \rangle \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product.

Polyhedra. In [LH98] a hat function $h(\mathbf{x})$ is constructed by the pointwise minimum of these tangents. We have

$$h(\mathbf{x}) = \min_{i=1,\dots,m} T^{-1}(\ell_i(\mathbf{x})) \tag{3}$$

The domains in which a particular tangent $\ell_i(\mathbf{x})$ determines the hat function are simple convex polyhedra P_i, which may be bounded or not (for details about convex polyhedra see [Grü67, Zie95]). Then a sweep-plane technique for generating random tuples in such a polyhedron with density proportional to $T^{-1}(\ell_i(\mathbf{x}))$ is derived.

To avoid lots of indices we write $\mathbf{p}$, $\ell(\mathbf{x})$ and P without the index i if there is no risk of confusion.

A sweep-plane algorithm. Let

$$\mathbf{g} = -\frac{\nabla \tilde{f}(\mathbf{p})}{\|\nabla \tilde{f}(\mathbf{p})\|} \tag{4}$$

if $\nabla \tilde{f}(\mathbf{p}) \neq 0$. Otherwise choose any $\mathbf{g}$ with $\|\mathbf{g}\| = 1$. ($\|\cdot\|$ denotes the 2-norm.) For a given $\mathbf{x}$ let $x = \langle \mathbf{g}, \mathbf{x} \rangle$. We denote the hyperplane perpendicular to $\mathbf{g}$ through $\mathbf{x}$ by

$$F(\mathbf{x}) = F(x) = \{\mathbf{y} \in \mathbb{R}^n : \langle \mathbf{g}, \mathbf{y} \rangle = x\} \tag{5}$$

and its intersection with the polytope P with $Q(\mathbf{x}) = Q(x) = P \cap F(x)$. ($F(\mathbf{x})$ depends on x only; thus we write $F(x)$, if there is no risk of confusion.) $Q(x)$ again is a convex simple polyhedra. Now we can move this sweep-plane $F(x)$ through the domain P by varying x. Figure 2 illustrates the situation.

As can easily be seen from (2), (4) and (5), $T^{-1}(\ell(\mathbf{x}))$ is constant on $Q(x)$ for every x. Let

$$\alpha = \tilde{f}(\mathbf{p}) - \langle \nabla \tilde{f}(\mathbf{p}), \mathbf{p} \rangle \quad \text{and} \quad \beta = \|\nabla \tilde{f}(\mathbf{p})\| \tag{6}$$

Then the hat function in P is given by

$$h|_P(\mathbf{x}) = T^{-1}(\ell(\mathbf{x})) = T^{-1}(\ell(x \cdot \mathbf{g})) = T^{-1}(\alpha - \beta\, x). \tag{7}$$

where again $x = \langle \mathbf{g}, \mathbf{x} \rangle$. We find for the marginal density function of the hat $h|_P$ along $\mathbf{g}$

$$h_{\mathbf{g}}(x) = \int_{Q(x)} h|_P(\mathbf{y})\, d\mathbf{y} = A(x) \cdot T^{-1}(\alpha - \beta\, x) \tag{8}$$

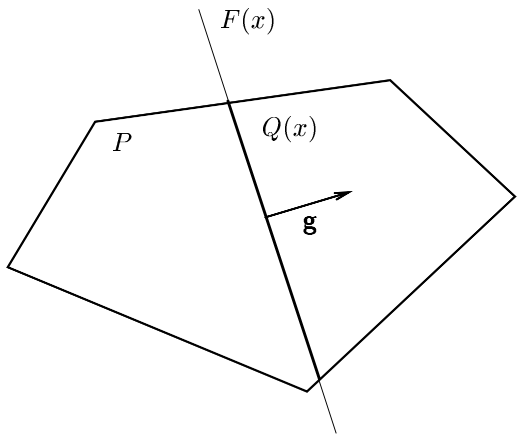

Figure 2: sweep-plane $F(x)$

where integration is done over $F(x)$. $A(x)$ denotes the $(n-1)$-dimensional volume of $Q(x)$. It exists if and only if $Q(x)$ is bounded.

To compute $A(x)$ let $\mathbf{v}_j$ denote the vertices of P and $v_j = \langle \mathbf{g}, \mathbf{v}_j \rangle$. Now assume that the polyhedron P is simple. Then let $\mathbf{t}_1^{\mathbf{v}_j}, \ldots, \mathbf{t}_n^{\mathbf{v}_j}$ be the n nonzero vectors in the directions of the edges of P originated from $\mathbf{v}_j$, i.e. for each k and every $\mathbf{x} \in P$, $\langle \mathbf{t}_k^{\mathbf{v}_j}, \mathbf{x} \rangle \geq 0$. Then by modifying the method in [Law91] we find

$$A(x) = \sum_{\substack{j \\ v_j \leq x}} a_j \, (x - v_j)^{n-1} = \sum_{k=0}^{n-1} b_k^{(x)} \, x^k \tag{9}$$

The coefficients are given by

$$a_j = \frac{1}{(n-1)!} \, | \det(\mathbf{t}_1^{\mathbf{v}_j}, \ldots, \mathbf{t}_n^{\mathbf{v}_j}) | \prod_{i=1}^{n} \langle \mathbf{g}, \mathbf{t}_i^{\mathbf{v}_j} \rangle^{-1} \tag{10}$$

and

$$b_k^{(x)} = \binom{n-1}{k} \sum_{\substack{j \\ v_j \leq x}} a_j \, (-v_j)^{n-1-k} \tag{11}$$

Notice that $b_k^{(x)} = b_k^{(v_{j-1})}$ for $x \in [v_{j-1}, v_j)$, and that equations (9) and (10) does not hold if P is not simple. For details see [LH98].

The generation from $h_{\mathbf{g}}$ is not easy in general. But for log-concave or T_c-concave (see §4.8) densities $f(\mathbf{x})$, $h_{\mathbf{g}}$ again is log-concave ([Pré73]) and T_c-concave ([LH98]), respectively.

Generate random tuples. For sampling from the “hat distribution” we first need the volume below the hat in all the polyhedra P_i and in the domain D.

We then choose one of these polytopes randomly with density proportional to their volumes. By means of a proper univariate random number we sample from marginal distribution $h|_{\mathbf{g}}$ and get a intersection $Q(x)$ of P. At last we have to sample from a uniform distribution on $Q(x)$.

It can be shown (see [LH98]) that the algorithm works if

(1) the polyhedra P_i are simple (see above),

(2) there exists a unique maximum of $\ell_i(\mathbf{x})$ in P_i (then $\alpha - \beta\, x$ is decreasing and thus the volume below the hat is finite in unbounded polyhedra), and

(3) $\ell_i(\mathbf{x})$ is non-constant on every edge of P_i (otherwise $\langle \mathbf{g}, \mathbf{t}_i^{\mathbf{v}_j} \rangle = 0$ for a vertex $\mathbf{v}_j$ and an edge $\mathbf{t}_i$ and thus $a_j = \infty$ in (10)).

Adaptive rejection sampling. It is very hard to find optimal points for constructing these tangents $\ell_i(\mathbf{x})$. Thus these points must be chosen by adaptive rejection sampling (see [GW92]). Adapted to our situation it works in the following way: We start with the $n + 1$ vertices of a regular simplex and add a new construction point whenever a point is rejected until the maximum number N of tangents is reached. The points of contact are thus chosen by a stochastic algorithm and it is clear that the multivariate density of the distribution of the next point for a new tangent is proportional to $h(\mathbf{x}) - f(\mathbf{x})$. Hence with N tending towards infinity the acceptance probability for a hat constructed in such a way converges to 1 with probability 1. It is not difficult to show that the expected volume below the hat is $1 + O(N^{-2/n})$.

Problems. Using this method we run into several problems.

- We have to compute the polyhedra every time we add a point.
- What must be done, if the marginal distribution (8) does not exist in the initial (usually not bounded) polyhedra P_i, or if the volume below the hat is infinite ($Q_i(x)$ not bounded, $\alpha - \beta\, x$ not decreasing)?

Moreover the polyhedra P_i typically have many vertices. Therefore the algorithm is slow and hard to implement because of the following effects.

- The computation of the polyhedra (setup) is very expensive.
- The marginal density (8) is expensive to compute. Since it is different for every polyhedron P_i (and for every density function f), we have to use a slow black box method (e.g. [GW92, Hör95a]) for sampling from the marginal distribution even in the case of log-concave densities.
- $Q(x)$ is not a simplex. Thus we have to use the (slow) recursive sweep-plane algorithm as described in [LH98] for sampling from the uniform distribution over a (simple) polytope.

2.3 Simple cones

A better idea is to choose the polyhedra first as simple as possible, i.e. we choose cones. (We describe in §2.4 how to get such cones.)

A *simple cone* C (with its vertex in the origin) is an unbounded subset spanned by n linearly independent vectors:

$$\begin{gathered} \mathbf{t}_1, \ldots, \mathbf{t}_n \in S^{n-1} \\ C = \{\lambda_1 \mathbf{t}_1 + \cdots + \lambda_n \mathbf{t}_n : \lambda_i \geq 0\} \end{gathered} \tag{12}$$

In opposition to the procedure described above we now have to choose a proper point $\mathbf{p}$ in this cone C for constructing a tangent. In the whole cone the hat h is then given by this tangent. The method itself remains the same.

Obviously the hat function is not continuous any more (because we first define a decomposition of the domain and then compute the hat function over the different parts. It cannot be made continuous by taking the pointwise minimum of the tangents, since otherwise we cannot compute the marginal density $h_{\mathbf{g}}$ by equation (8)). Moreover we have to choose one touching point in each part. These disadvantages are negligible compared to the enormous speedup of the setup and of the generation of random tuples with respect to this hat function.

Marginal density. The intersection $Q(x)$ of the sweep plane $F(x)$ with the cone C is bounded if and only if $F(x)$ cuts each of the sets $\{\lambda \mathbf{t}_i : \lambda > 0\}$ for all $x > 0$, i.e. if and only if $\langle \mathbf{g}, \lambda \mathbf{t}_i \rangle = x$ for a $\lambda > 0$ by (5), and hence if and only if

$$\langle \mathbf{g}, \mathbf{t}_i \rangle > 0 \qquad \text{for all } i. \tag{13}$$

We find for the volume $A(x)$ in (9) of the intersection $Q(x)$

$$A(x) = \begin{cases} a\, x^{n-1} & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{14}$$

where (again)

$$a = \frac{1}{(n-1)!} \,|\det(\mathbf{t}_1, \ldots, \mathbf{t}_n)|\, \prod_{i=1}^{n} \langle \mathbf{g}, \mathbf{t}_i \rangle^{-1} \tag{15}$$

Notice that $A(x)$ does not exist if condition (13) is violated, whereas the right hand side in (14) is defined if $\langle \mathbf{g}, \mathbf{t}_i \rangle \neq 0$ for all i.

If the marginal density exists, i.e. (13) holds, then by (8) and (6) it is given by

$$h_{\mathbf{g}}(x) = a\, x^{n-1}\, T^{-1}(\alpha - \beta\, x) \tag{16}$$

Volume. The volume below the hat function in a cone C is given by

$$H_C = \int_0^\infty h_{\mathbf{g}}(x)\, dx = \int_0^\infty a\, x^{n-1}\, T^{-1}(\alpha - \beta\, x)\, dx \tag{17}$$

Notice that $\mathbf{g}$ and thus a, α and β depend on the choice of $\mathbf{p}$. Choosing an arbitrary $\mathbf{p}$ may result in a very large volume below the hat and thus in a very poor rejection constant.

Intersection of sweep-plane. Notice that the intersection $Q(x)$ is always a $(n-1)$-simplex if condition (13) holds. Thus we can use the algorithm in [Dev86] for sampling from uniform distribution on $Q(x)$. The vertices $\mathbf{v}_1, \dots, \mathbf{v}_n$ of $Q(x)$ in $\mathbb{R}^n$ are given by

$$\mathbf{v}_i = \frac{x}{\langle \mathbf{g}, \mathbf{t}_i \rangle} \, \mathbf{t}_i \tag{18}$$

Let U_i $(i = 1, \dots, n-1)$ iid uniformly $[0,1]$ random variates and $U_0 = 0$, $U_n = 1$. We sort these variates such that $U_0 \leq \dots \leq U_n$. Then we get a random point in $Q(x)$ by (see [Dev86, theorems XI.2.5 and V.2.1])

$$\mathbf{X} = \sum_{i=1}^{n} (U_i - U_{i-1}) \, \mathbf{v}_i \tag{19}$$

The choice of p. One of the main difficulties of the new approach is the choice of the touching point $\mathbf{p}$. In opposite to the first approach where the polyhedron is build around the touching point, we now have to find such a point so that (13) holds. Moreover the volume below the hat function over the cone should be as small as possible.

Searching for such a touching point in the whole cone C or in domain D (the touching point needs not to be in C) with techniques for multidimensional minimization is not very practicable. Firstly the evaluation of the the volume H_C in (17) for a given point $\mathbf{p}$ is expensive and its gradient with respect to $\mathbf{p}$ is not given. Secondly the domain of H_C is given by the set of points where (13) holds.

Instead we suggest to choose a point in the center of C for a proper touching point for our hat. Let $\bar{\mathbf{t}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{t}_i$ be the barycenter of the spanning vectors. Let $a(s)$, $\alpha(s)$ and $\beta(s)$ denote the corresponding parameters in (16) for $\mathbf{p} = s\,\bar{\mathbf{t}}$. Then we choose $\mathbf{p} = s\,\bar{\mathbf{t}}$ by minimizing the function

$$\eta \colon \mathcal{D}_A \to \mathbb{R}, \quad s \mapsto \int_0^\infty a(s)\, x^{n-1}\, T^{-1}(\alpha(s) - \beta(s)\, x)\, dx \tag{20}$$

The domain $\mathcal{D}_A$ of this function is given by all points, where $\|\nabla \tilde{f}(s\bar{\mathbf{t}})\| \neq 0$ and where $A(x)$ exists, i.e. where $\mathbf{g} = \mathbf{g}(s\,\bar{\mathbf{t}})$ fulfils condition (13). It can easily be seen, that $\mathcal{D}_A$ is an open subset of $(0, \infty)$.

To minimize η we can use standard methods, e.g. Brent's algorithm (see e.g. [FMM77]). The main problem is to find $\mathcal{D}_A$. Although $\tilde{f}(\mathbf{x})$ is concave by assumption, it is possible for a particular cone C that $\mathcal{D}_A$ is a strict subset of $(0, \infty)$ or even the empty set. Moreover it might not be connected. In general only the following holds: Let (a, b) be a component of $\mathcal{D}_A \neq \emptyset$. If If $f \in \mathcal{C}^1$, i.e. the gradient of f is continuous, then

$$\lim_{s \searrow a} \eta(s) = \infty \qquad \text{and} \qquad \lim_{s \nearrow b} \eta(s) = \infty \tag{21}$$

Roughly spoken, η is a U-shaped function on (a, b).

An essential part of the minimization is initial bracketing of the minimum, i.e. finding three points $s_0 < s_1 < s_2$ in (a, b), such that $\eta(s_1) < \eta(s_0)$ and $\eta(s_1) < \eta(s_2)$. This is necessary since the function term of η in (20) is also defined for some $s \notin \mathcal{D}_A$ (e.g. $s < 0$). Using Brent's algorithm without initial bracketing may (and occasionally does) result in e.g. a negative s.

Bracketing can be done by (1) search for a $s_1 \in \mathcal{D}_A$, and (2) use property (21) and move towards a and b, respectively, to find an s_0 and an s_2. (It is obvious that we only find a local minimum of η by this procedure. But in all the distributions we have tested, there is just one local minimum which therefore is the global one.)

For the special case where $\langle \mathbf{g}(s), \bar{\mathbf{t}} \rangle$ does not depend on s (e.g. for all multivariate normal distributions) $\mathcal{D}_A$ either is $(0, \infty)$ or the empty set. It is then possible to make similar considerations like that in [Hör95a, theorem 2.1] for the one dimensional case. Adapted to the multivariate case it would state, that for the optimal touching point $\mathbf{p}$, $f(\mathbf{p})$ is the same for every cone C.

Condition violated. Notice that $\mathcal{D}_A$ even may be the empty set, i.e. condition (13) fails for all $s \in (0, \infty)$. By the concavity of $\tilde{f}(\mathbf{x})$ we know, that $\langle \mathbf{g}, \mathbf{p} \rangle > 0$ for every construction point $\mathbf{p}$. Furthermore $\langle \mathbf{g}, \mathbf{p} \rangle$ is bounded from below on every compact subset of the domain D of the density f. Therefore there always exists a partition into simple cones with proper touching points $\mathbf{p} = s\,\bar{\mathbf{t}}$ which satisfy (13), i.e. the domains $\mathcal{D}_A$ are not empty for all cones C. We even can have $\mathcal{D}_A = (0, \infty)$.

2.4 Triangulation

For this new approach we need a partition of the $\mathbb{R}^n$ into simple cones. We get such a partition by triangulation of the unit sphere S^{n-1}. Each cone C is then generated by a simplex $\Delta \subset S^{n-1}$ (triangle in S^2, tetrahedron in S^3, and so on):

$$C = \{\lambda\,\mathbf{t} \colon \lambda \geq 0,\, \mathbf{t} \in \Delta\} \tag{22}$$

These simplices are uniquely determined by the vectors $\mathbf{t}_1, \ldots, \mathbf{t}_n$ in (12), i.e. their vertices. (They are the the convex hull of these vertices in S^{n-1}.) It does not matter that these cones are closed sets. The intersection of such cones might not be empty but has measure zero.

For computing a in (15) we need the volumes of these simplices. To avoid $\mathcal{D}_A$ being the empty set, some of the cones have to be skinny. Furthermore to get a good hat function, these simplices should have the same volume (if possible) and they should be "regular", i.e. the distances from the center to the vertices should be equal (or similar). Thus the triangulation should have the following properties:

(C1) Recursive construction.

(C2) $|\det(\mathbf{t}_1, \ldots, \mathbf{t}_n)|$ are easy computable for all simplices.

(C3) Edges of a simplex have equal length.

Although it is not possible to get such a triangulation for $n \geq 3$ we suggest an algorithm which fulfils (C1) and (C2) and which "nearly" satisfies (C3).

Initial cones. We get the initial simplices as the convex hull in S^{n-1} of the vectors

$$\delta_1 \mathbf{e}_1, \ldots, \delta_n \mathbf{e}_n \tag{23}$$

where $\mathbf{e}_i$ denotes the i-th unit vector in $\mathbb{R}^n$ (i.e. a vector where the i-th component is 1 and all others are 0) and $\delta_i \in \{-1, 1\}$. As can easily be seen the resulting partition of the $\mathbb{R}^n$ is that of the arrangement of the hyperplanes

$$x_i = 0, \qquad i = 1, \ldots, n \tag{24}$$

Hence we have 2^n initial cones.

Barycentric subdivision of edges. To get smaller cones we have to triangulate these simplices. Standard triangulations of simplices which are used for example in fixed-point computation (see e.g. [Tod76, Tod78]) are not appropriate for our purpose. The number of simplices increases too fast for each triangulation step. (In opposition to fixed point calculations, we have to keep all simplices with all their parameters in the memory of the computer.)

Instead we use a barycentric subdivision of edges: Let $\mathbf{t}_1, \ldots, \mathbf{t}_n$ be the vertices of a simplex Δ. Then use the following algorithm.

(1) Find the longest edge $(\mathbf{t}_i, \mathbf{t}_j)$.

(2) Let

$$\mathbf{t}_{new} = \frac{\mathbf{t}_i + \mathbf{t}_j}{\|\mathbf{t}_i + \mathbf{t}_j\|}, \tag{25}$$

i.e. the barycenter of the edge projected to the sphere.

(3) Get two smaller simplices: Replace vertex $\mathbf{t}_i$ by $\mathbf{t}_{new}$ for the first simplex and vertex $\mathbf{t}_j$ by $\mathbf{t}_{new}$ for the second one. We have

$$|\det(\mathbf{t}_1, \ldots, \mathbf{t}_{new}, \ldots, \mathbf{t}_n)| = \frac{1}{\|\mathbf{t}_i + \mathbf{t}_j\|} \, |\det(\mathbf{t}_1, \ldots, \mathbf{t}_n)| \tag{26}$$

After making k of such triangulation steps in all initial cones we have 2^{n+k} simplices.

This triangulation is more flexible. Whenever we have a cone C, where $\mathcal{D}_a$ is empty (or the algorithm does not find an $s \in \mathcal{D}_a$) we can split C and try again to find a proper touching point in both new cones. This can be done until we have found proper construction points for all cones of the partition (see end of §2.3). In practice this procedure stops, if too many cones are necessary. (The computer runs out of memory.)

Notice that it is not a good idea to use barycentric subdivision of the whole simplex (instead of dividing the longest edge). This triangulation exhibits the inefficient behavior of creating long, skinny simplices (see remark in [Tod76]).

"Oldest" edge. Finding the longest of the $\binom{n}{2}$ edges of a simplex is very expensive. An alternative approach is to use the "oldest" edge of a simplex. The idea is the following:

(1) Enumerate the $2n$ vertices of the initial cones.

(2) Whenever a new vertex is created by barycentric subdivision, it gets the next number.

(3) Edges are indexed by the tuple (i, j) of the number of the incident vertices, such that $i < j$.

(4) We choose the edge with the lowest index with respect to the lexicographic order (the "oldest" edge). This is just the pair of lowest indices of the vertices of the simplex.

As can easily be seen, the "oldest" edge is (one of) the longest edge(s) for the first $n - 1$ triangulation steps. Unluckily this does not hold for all simplices in following triangulation steps. (But it is at least not the shortest one.)

Computational experiences with several normal distributions for some dimensions n have show, that this idea speeds up the triangulation enormously but has very little effect on the rejection constant.

Setup. The basic version of the setup algorithm is as follows:

1. Create initial cones.
2. Triangulate.
3. Find touching points $\mathbf{p}$ if possible (and necessary).
4. Triangulate every cone without proper touching point.
5. Goto 3 if cones without proper touching points exist, otherwise stop.

2.5 Problems

Although this procedure works for our tested distributions, an adaptation might be necessary for a particular density function f.

(1) The searching algorithm for a proper touching point in §2.3 can be improved. E.g. $\mathcal{D}_A$ is either $[0, \infty)$ or the empty set if f is a normal distribution.

(2) There is no criterion how many triangulation steps are necessary or usefull for an optimal rejection constant. Thus some tests with different numbers of trianglation steps should be made with density f (see also §5).

(3) It is possible to triangulate each cone with a "bad" touching point. But besides the case where no proper touching point can be found, some touching points may lead to an enourmous volume below the hat function. So this case should also be excluded and the corresponding cones should be triangulated.

A simple solution to this problem is that an upper bound $H_{\max}$ for the volumes H_C is provided. Each cone with $H_C > H_{\max}$ has to be triangulated further. Such a bound can be found by some empirical tests with the given density f.

Another way is to triangulate all initial cones first and then let $H_{\max}$ be a multiple (e.g. 10) of the 90th percentile of the H_C of all created cones.

(4) Problems might occur when the mode is on the boundary of the support $\mathrm{supp} f = \{\mathbf{x} \in D : f(\mathbf{x}) \neq 0\} \subset \mathbb{R}^n$. (Then we set $\log(f(\mathbf{x})) = -\infty$ for all $\mathbf{x} \in D \setminus \mathrm{supp} f$ and thus $\log \circ f : D \to \mathbb{R} \cup \{-\infty\}$ can be seen as a concave function.) An example for such a situation is when $f(\mathbf{x})$ is a normal density on a ball B and vanishes outside of B.

In such a case there exists a cone C such that $\{\lambda \bar{\mathbf{t}} : \lambda > 0\}$ does not intersect $\mathrm{supp} f$ and the algorithm is in troubles. If $C \cap \mathrm{supp} f = \emptyset$ we simply can remove this cone. Otherwise an expensive search for a proper touching point is necessary.

Restrictions. The above observations — besides the fact that no automatic adaption is possible — are a drawback of the algorithm for its usage as black-box algorithm. Nevertheless the algorithm is suitable for a large class of log-concave densities and it is possible to include parameters into the code to adjust the algorithm for a given density easily. Of course some tests might be necessary. Besides, the algorithm does not produce wrong random points but simply does not work, if no "good" touching points can be found for some cones C.

2.6 Log-concave densities

The transformation $T(x) = \log(x)$ satisfies (T1)–(T4). If $T(f(\mathbf{x})) = \log(f(\mathbf{x}))$ is concave, we say f is *log-concave*. We have $T^{-1}(x) = \exp(x)$ and thus we find for the marginal density function in (16) those of a gamma distribution with shape parameters n and β.

$$h_{\mathbf{g}}(x) = a\, x^{n-1} \exp(\alpha - \beta\, x) = a\, e^{\alpha} \cdot x^{n-1}\, e^{-\beta\, x} \tag{27}$$

The volume below the hat for log-concave densities in a cone C is now given by

$$H_C = \int_0^\infty a\, x^{n-1} \exp(\alpha - \beta\, x)\, dx = a\, e^{\alpha}\, \beta^{-n}\, (n-1)! \tag{28}$$

To minimize this function it is best to use its logarithm:

$$\mathcal{D}_A \to \mathbb{R}, \quad s \mapsto \log(a(s)) + \alpha(s) - n\, \log(\beta(s)) \tag{29}$$

For the normal distribution with density proportional to $f(\mathbf{x}) = \exp(-\sum x_i^2)$ we have $\tilde{f}(\mathbf{p}) = \tilde{f}(s\bar{\mathbf{t}}) = \log(f(s\bar{\mathbf{t}})) = -\sum s^2\, \bar{t}_i^2 = -s^2$, where $\bar{\mathbf{t}}$ is the center of the cone C with $\sum \bar{t}_i^2 = 1$. Thus we simply find by (6) $\alpha(s) = s^2$ and $\beta = 2s$. Since $a(s)$ does not depend on s we find for (29) $s^2 - n\log(s) + \text{constant}$. But even for the normal distribution with an arbitrary covariance matrix, this function becomes much more complicated.

3 The algorithm

The algorithm TDRMV() consists of two main parts: the construction of a hat function $h(\mathbf{x})$ and the generation of random tuples $\mathbf{X}$ with density proportional to this hat function. The first one is done by the subroutine SETUP(), the second one by the routine SAMPLE().

Algorithm 2 TDRMV() /* generate a random tuple for given log-concave density */

Input: density f

/* Setup */
call SETUP(). /* Construct a hat-function $h(\mathbf{x})$ */
/* Generator */
repeat
  $\mathbf{X} \leftarrow$ **call** SAMPLE(). /* Generate a random tuple $\mathbf{X}$ with density prop. to $h(\mathbf{X})$. */
  Generate a uniform random number U.
until $U \cdot h(\mathbf{X}) \le f(\mathbf{X})$.
return $\mathbf{X}$.

To store $h(\mathbf{x})$, we need a list of all cones C. For each of these cones we need several data which we store in the object cone. Notice that the variables $\mathbf{p}$, $\mathbf{g}$, α, β, a and H_C depend on the choice the touching point $\mathbf{p}$ and thus on s. Some of the parameters are only necessary for the setup.

object 1 cone

PARAMETER	VARIABLE	DEFINITION	
spanning vectors	$\mathbf{t}_1, \dots, \mathbf{t}_n$		
center of cone	$\bar{\mathbf{t}}$	$= \frac{1}{n}\sum \mathbf{t}_i$	(setup)
construction point	$\mathbf{p}$	$= s \cdot \bar{\mathbf{t}}$	(setup)
location of $\mathbf{p}$	s		(setup)
sweep plane	$\mathbf{g}$	see (4)	
marginal density	α, β	see (6)	
coefficient	a	see (15)	
determinant of vectors	det	$= \det(\mathbf{t}_1, \dots, \mathbf{t}_n)$	(setup)
volume under hat	H_C, H_C^{cum}	see (17) and (28)	

Remark. To make the description of the algorithm more readable, some standard techniques are not given in details.

3.1 Setup

The routine SETUP() consists of three parts: (H1) setup initial cones, (H2) triangulation of the initial cones and (H3) calculation of parameters.

(H1) is simple (see §2.4). (H2) is done by subroutine SPLIT(). The main problem in (H3) is how to find the parameter s (i.e. a proper construction point). This is done by subroutine FIND(). Minimizing (29) is very expensive. Notice that for a given s we have to compute all parameters that depend on s before evaluating this function. Since it is not suitable to use the derivative of this function, a good choice for finding the minimum is to use Brent's algorithm (e.g. [FMM77]). To reduce the cost for finding a proper s, we do not minimize (29) for every cone. Instead we use the following procedure:

(1) Make some triangulation steps as described in §2.4.

(2) Compute s for every cone C.

(3) Continue with triangulation. When a cone is split by barycentric subdivision of the corresponding simplex, both new cones inherit s from the old simplex.

Our computational experiences with various normal distributions show, that the costs for setup reduces enormously without raising the rejection constant too much. Using this procedure it might happen that s does not give a proper touching point $\mathbf{p} = s\,\bar{\mathbf{t}}$ (or H_C is too big; see end of §2.4) after finishing all triangulation steps. Thus we have to check s for every cone and continue with triangulation in some cones if necessary.

3.2 Sampling

The subroutine SAMPLE() consists of four parts: (S1) select a cone C, (S2) find a random variate proportional to the marginal density $h_{\mathbf{g}}$ (27), (S3) generate a uniform random tuple $\mathbf{U}$ on the standard simplex (i.e. $0 \leq U_1 \leq \ldots \leq U_n \leq 1$ and $U_1 + \cdots + U_n = 1$) and (S4) compute tuple on the intersection $Q(x)$ of the sweeping plane with cone C. (S3) and (S4) is done by subroutine SIMPLEX().

4 Possible variants

4.1 Subset of $\mathbb{R}^n$ as domain

Our experiments have shown, that the basic algorithm works even for densities with support $\operatorname{supp} f = \{\mathbf{x} \in D\colon f(\mathbf{x}) \neq 0\} \subset \mathbb{R}^n$. Since the hat $h(\mathbf{x})$ has support $\operatorname{supp} h = \mathbb{R}^n$, the rejection constant might become very big.

Subroutine 3 SETUP() /* Construct a hat function */

Input: level of triangulation for finding s, level of minimal triangulation

/* Initial cones */
for all tuples $(\delta_1, \ldots, \delta_n)$ with $\delta_i \in \{\pm 1\}$ **do** /* 2^n initial cones */
    Append new cone to list of cones with $\delta_1 \mathbf{e}_1, \ldots, \delta_n \mathbf{e}_n$ as its spanning vectors.
/* Triangulation */
repeat
    for all cone C in list of cones **do**
        call SPLIT() **with** C.
    Update list of cones.
until level of triangulation for finding s is reached
/* Find s */
for all cone C in list of cones **do**
    call FIND() **with** C.
/* Continue triangulation */
repeat
    for all cone C in list of cones **do**
        call SPLIT() **with** C.
    Update list of cones.
until minimum level of triangulation is reached
/* Check s */
repeat
    for all cone C in list of cones where s unknown **do** /* (13) violated */
        call SPLIT() **with** C and list of cones.
        call FIND() **with** both new cones.
    Update list of cones.
until no such cone was found
/* Compute all parameters */
for all cone C in list of cones **do**
    Compute all parameters of C.
/* Total volume below hat */
$H_{tot} \leftarrow 0$.
for all cone C in list of cones **do**
    $H_{tot} \leftarrow H_{tot} + H_C$.
    $H_C^{cum} \leftarrow H_{tot}$ /* Used for $O(0)$-search algorithm */
/* End */
return list of cones, H_{tot}.

Subroutine 4 SPLIT() /* split a given cone and update list of cones */

Input: cone C, list of cones

Find lowest indices i, j of all vectors of C.
Find highest index m of all vectors (of triangulation).
$\mathbf{t}_{m+1} \leftarrow \frac{\mathbf{t}_i + \mathbf{t}_j}{\|\mathbf{t}_i + \mathbf{t}_j\|}$.
Append new cone C' to list and copy vectors and s of C into C'.
Replace $\mathbf{t}_i$ by $\mathbf{t}_{m+1}$ in C and replace $\mathbf{t}_j$ by $\mathbf{t}_{m+1}$ in C'.
Replace det by $\frac{1}{\|\mathbf{t}_i + \mathbf{t}_j\|} \cdot \det$ in C and C'.
return list of cones.

Subroutine 5 FIND() /* find a proper touching point */

Input: cone C

/* Bracketing a minimum */
Search for a $s_1 \in \mathcal{D}_A$. **return** `failed` if not successful.
Search for s_0, s_2 (Use property (21)). **return** `failed` if not successful.
/* Find minimum */
Find s using Brent's algorithm (Use (29)). **return** `failed` if not successful.

Subroutine 6 SAMPLE() /* Generate a random tuple with density proportional to hat */

Input: H_{tot}, list of cones

/* Find cone */
Generate a uniformly $[0, H_{tot}]$ distributed random variate U.
Find C, such that $H^{cum}_{C_{pred}} \leq U < H^{cum}_{C}$.
(C_{pred} is the predecessor of C is the list of cones.)
/* Find sweep-plane */
Generate a gamma(n, β) distributed random variate G.
/* Generate uniformly distributed point in $Q(G)$ and return tuple */
$\mathbf{X} \leftarrow$ **call** SIMPLEX() **with** C and G.
return $\mathbf{X}$.

Subroutine 7 SIMPLEX() /* Generate a uniform distributed tuple on simplex */

Input: cone C, x (location of sweeping plane)

/* Generate n uniformly distributed random variates U_i in simplex */
Generate iid uniform $[0,1]$ random variates U_i, $i = 1, \dots, n-1$.
$U_n \leftarrow 1$.
Sort: $0 \leq U_1 \leq \dots \leq U_{n-1} \leq U_n = 1$.
for $i = n, \dots, 2$ **do**
    $U_i \leftarrow U_i - U_{i-1}$.
/* Generate uniformly distributed point $\mathbf{X}$ in $Q(x)$ */
$\mathbf{X} \leftarrow \sum_{i=1}^{n} U_i \frac{x}{\langle \mathbf{g}, \mathbf{t}_i \rangle} \mathbf{t}_i$.
/* Return random tuple */
return $\mathbf{X}$.

Pyramids. If the given domain D is a proper subset of $\mathbb{R}^n$ (that is, we give constraints for $\mathrm{supp} f$), the acceptance probability can be increased when we restrict the domain of h accordingly to the domain D. (The domain is the set of points where the density f is defined; obviously $\mathrm{supp} f \subseteq D$. Notice that we have to provide the domain D for the algorithm but the support of f is not known.)

Thus we replace (some) cones by pyramids. Notice that the base of such a pyramid must be perpendicular to the direction $\mathbf{g}$. Hence we first have to choose a construction point $\mathbf{p}$ and then compute the height of the pyramid.

The union of these pyramids (and of the remaining cones) must cover D. Whenever we get a random point not in the domain D we reject it. It is clear that continued triangulation decreases the volume between D and enclosing set.

Polytopes. We only deal with the case where D is an arbitrary polytope which are given by a set of linear inequalities.

Height of pyramid. The height is the maximum of $\langle \mathbf{g}, \mathbf{x} \rangle$ in $C \cap D$. Because of our restriction to polytopes this is a linear programming problem. Using the spanning vectors $\mathbf{t}_1, \ldots, \mathbf{t}_n$ as basis for the $\mathbb{R}^n$, it can be solved by means of the simplex algorithm in at most k pivot steps (for a simple polytope), where k is the number of constraints for D.

Marginal density and volume below hat. The marginal distribution is a truncated gamma distribution with domain $[0, u]$, where u is the height of the pyramid C. Instead of (28) and (29) we find for pyramids

$$H_C = \int_0^u x^{n-1} \exp(\alpha - \beta\, x)\, dx = a\, e^{\alpha}\, \beta^{-n}\, \Gamma(n, \beta u) \tag{30}$$

and

$$\mathcal{D}_A \to \mathbb{R}, \quad s \mapsto \log(a(s)) + \alpha(s) - n\, \log(\beta(s)) + \log(\Gamma(n, \beta(s) u(s))) \tag{31}$$

where $\Gamma(n, x) = \int_0^x t^{n-1} e^{-t}\, dt$ is proportional to the incomplete gamma function and can be computed by means of formula (3.351) in [GR65].

Computing the height $u(s)$ is rather expensive. So it is recommended to use (29) instead of the exact function (31) for finding a touching point in pyramid C. Computational experiments with the standard normal distribution have shown, that the effect on the rejection constant is rather small (less than 5%).

4.2 Density not differentiable

For the construction of the hat function we need a tangent plane for every $\mathbf{x} \in D$. Differentiability of the density is not really required. Thus it is sufficient to have a subroutine that returns the normal vector of a tangent hyperplane (which is "$\nabla f(\mathbf{x})$", if $f \in \mathcal{C}^1$) for every $\mathbf{x}$.

However for densities f which are not differentiable the function in (29) might have a nasty behavior. However notice that f must be continuous in the interior of supp$f$, since $log \circ f$ is concave.

4.3 Indicator Functions

If $f(\mathbf{x}) = f_0$ is the indicator function of a convex set, then we can choose an arbitrary point in the convex set as the mode (as origin of our construction) and set $\mathbf{g} = \bar{\mathbf{t}}$, the center of the cone C (see (4) in §2.2). Notice that the marginal density in (16) now reduces to $h_{\mathbf{g}}(x) = a\, f_0\, x^{n-1}$. None of the parameters α and β depends on the choice of the touching point $\mathbf{p}$. Of course we have to provide a compact domain for the density.

Using indicator functions we can generate uniformly distributed random variates of arbitrary convex sets.

4.4 Mode not in Origin

It is obvious that the method works, when the mode $\mathbf{m}$ is an arbitrary point in $D = \mathbb{R}$. If the mode is unknown we can use common numerical methods for finding the maximum of f, since $T(f(\mathbf{x}))$ is concave (see e.g. [Rao84]).

Notice that the exact location of the mode is not really required. The algorithm even works if the center for the construction of the cones is not close to the mode. Then we just get a hat with a worse rejection constant.

4.5 Add mode as construction point

Since we have only one construction point in each cone, the rejection constant is bounded from below. Thus only a few steps to triangulate the S^{n-1} make sense. To get a better hat function we can use the mode $\mathbf{m}$ of f as an additional construction point. The hat function is then the minimum of $f(\mathbf{m})$ and the original hat. The cone is split into two parts by a hyperplane $F(b)$ with different marginal densities, where b is given by $T^{-1}(\alpha - \beta\, b) = f(\mathbf{m})$. Its marginal density is then given by

$$h_{\mathbf{g}}(x) = a\, x^{n-1} \cdot \begin{cases} f(\mathbf{m}) & \text{for } x \le b \\ T^{-1}(\alpha - \beta\, x)) & \text{for } x > b \end{cases} \tag{32}$$

Notice that we use the same direction $\mathbf{g}$ for the sweep plane in both parts. We have to compute the volume below the hat for both parts which are given by

$$H_C^1 = a\, b^n \qquad \text{and} \qquad H_C^2 = \int_b^\infty a\, x^{n-1}\, T^{-1}(\alpha - \beta\, x)\, dx \tag{33}$$

4.6 More construction points per cone

A way to improve the hat function is to use more than one (or two) construction points. But this method has some disadvantages and it is not recommended to use it.

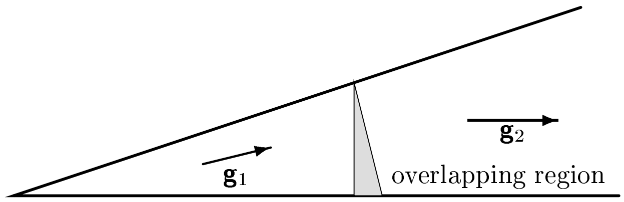

Figure 3: Two construction points in a cone

The cones are divided in several pieces of a pyramid (see figure 3). The lower and upper base of these pieces must be perpendicular to the corresponding direction **g**. These vectors **g** are determined by the gradients of the transformed density at the construction points in this pieces. Thus these **g** (may) differ and hence these pieces must overlap. This increases the rejection constant. Moreover it is not quite clear how to find such pieces. For the univariate case appropriate methods exist (e.g. [DH94]). But in the multivariate case these are not suitable.

Also adaptive rejection sampling (introduced in [GW92]) as used in [Hör95b, LH98] is not a really good choice. The reason is quite simple. The cones are fixed and the construction points always are settled in the center of these cones. Thus using adaptive rejection sampling we select the new construction points due to a distribution which is given by the marginal density of $(h-f)|_C$. And this marginal density is not zero at the existing construction points.

4.7 Squeezes

We can make a very simple kind of squeezes: Let $x_0 = 0 < x_1 < x_2 < \ldots < x_k$. Compute the minima of the transformed density at $Q(x_i)$ for all i. Since $\tilde{f}$ is concave these minima are at the vertices of these simplices. The squeeze $s_i(x)$ for $x_{i-1} \le x \le x_i$ is then given by

$$s_i(x) = T^{-1}\left(\frac{\bar{f}(x_i) - \bar{f}(x_{i-1})}{x_i - x_{i-1}}(x - x_{i-1}) + \bar{f}(x_{i-1})\right) \tag{34}$$

where $\bar{f}(x_i)$ denotes the minimum of $\tilde{f}(\mathbf{x})$ in $Q(x_i)$. The setup of these squeezes is rather expensive and only useful, if many random points of the same distribution must be generated.

4.8 T_c-concave densities

A family T_c of transformations that fulfil conditions (T1)–(T4) is introduced in [Hör95a]. Let $c \le 0$. Then we set

c		$T_c(x)$	$T_c^{-1}(x)$	$T_c'(x)$
$c = 0$	$\mathbb{R}^+ \to \mathbb{R}$	$\log(x)$	$\exp(x)$	x^{-1}
$-\frac{1}{n} < c \le 0$	$\mathbb{R}^+ \to \mathbb{R}^-$	$-x^c$	$(-x)^{1/c}$	$-c\,x^{c-1}$

It can easily be verified, that condition (T4) (i.e. volume below hat is bounded) holds if and only if $-\frac{1}{n} < c \leq 0$. Moreover for $c < 0$ we must have $T_c(h(\mathbf{x}))|_C < 0$. To ensure the negativity of the transformed hat we always have to choose the mode $\mathbf{m}$ as construction point (see §4.5).

In [Hör95a] it was shown that if a density f is T_c-concave then it is T_{c_1}-concave for all $c_1 \leq c$. The case $c = 0$, $T_0(x) = \log(x)$ is already described in §2.6.

For the case $c < 0$ the marginal density function (16) is now given by

$$h_{\mathbf{g}}(x) = a\,x^{n-1} \cdot \begin{cases} f(\mathbf{m}) & \text{for } x \leq b \\ (\beta\,x - \alpha)^{\frac{1}{c}} & \text{for } x > b \end{cases} \tag{35}$$

where b is given by $(\beta\,b - \alpha)^{\frac{1}{c}} = f(\mathbf{m})$. To our knowledge no special generator for this distribution is known. (The part for $x > b$ looks like a beta-prime distribution (see [JKB95]), but $\alpha, \beta > 0$.).

By assumption $(\beta\,x - \alpha) > 0$ for $x > b$ and $\frac{1}{c} < -n$. Hence it can easily be seen that the marginal density is T_c-concave. Therefore we can use the universal generator ([Hör95a]).

5 Computational Experience

5.1 A C-implementation.

A test version of the algorithm was written in C and is available via anonymous ftp [Ley98].

It should handle the following densities f:

- f is log-concave but not constant on its support.
- Domain D is either $\mathbb{R}^n$ or an arbitrary rectangle $[a_1, b_1] \times \ldots \times [a_n, b_n]$.
- The mode $\mathbf{m}$ is arbitrary. But if $D \neq \text{supp} f$ then $\mathbf{m}$ must be an interior point of $\text{supp} f$ not "too close" to the boundary of $\text{supp} f$.

We used two lists for storing the spanning vectors and the cones (with pointers to the list of vectors). For the setup we have to store the edges (i, j) for computing the new vertices. This is done temporarily in a hash table, where the first index i is used as the hash index.

The setup step is modified in the case of a rectangular domain. If the mode is near to the boundary of D we use the nearest point on the boundary (if possible a vertex) for the center to construct the cones. If this point is on the boundary we easily can eliminate all those initial cones, that does not intersect D. If this point is a vertex of the rectangle there remains only one initial cone.

For finding the mode of f we used a pattern search method by Hooke and Jeeves [HJ61, Rao84] as implemented in [Kau63, BP66], since it could deal with both unbounded and bounded support of f without giving explicit constraints. For finding the minimum of (29) we use Brent's algorithm as described in ([FMM77]). The implementation contains some parameters to adjust these routines to a particular density f.

For finding a cone C in subroutine SAMPLE() we used a $O(0)$-algorithm with a search table. (Binary search is slower.) For generating the gamma distributed random number G we used the algorithm in [AD82] for the case of unbounded domain. When D is a rectangle, we used transformed density rejection ([Hör95a]) to generate from the truncated gamma distributions. Here it is only necessary to generate a optimal hat function for the truncated gamma distribution with shape parameters n and 1 with domain $(0, u_{\max})$, where $u_{\max}$ is the maximal value of $height \cdot \beta$ for all cones. The optimal touching points for this gamma distribution are computed by means of the algorithm [DH94].

The code was written for testing various variants of the algorithm and is not optimized for speed. Thus the data shown in the tables below give just an idea of the performance of the algorithm.

We have tested the algorithm with various multivariate log-concave distributions in some dimensions. All our tests have been done on a PC with a P90 processor running Linux and the GNU C compiler.

5.2 Basic version: unbounded domain, mode in origin

Random points with density proportional to hat function. The time for the generation of random points below the hat has shown to be almost linear in dimension n. Table 1 shows the average time for the generation of a single point. For comparison we give the time for generation of n normal distributed points using the Box/Muller method [BM58] (which gives a standard multinormal distributed point with density proportional to $f(\mathbf{x}) = \exp(-\sum_{i=1}^{n} x_i^2))$. For computing the hat function we only used initial cones for the standard multinormal density.

n	2	3	4	5	6	7	8	9	10
hat function	14.6	17.1	21.3	24.9	30.2	34.6	41.5	45.7	55.6
multinormal	7.2	10.8	14.4	18.0	21.6	25.2	28.8	32.4	36.0

Table 1: average time for the generation of one random point (in μs)

Random points for the given density. The real time needed for the generation of a random point for a given log-concave density depends on the rejection constant and the costs for computing the density. Table 2 shows the acceptance probabilities and the times needed for the generation of standard multinormal distributed points. Notice that these data do not include the time for setting up the hat function.

Setup. When FIND() is called *after* triangulation has been done, the time needed for the computation of the hat function depends linearly on the number

n	2	3	4	5	6	7	8	9	10
number of cones	2^5	2^8	2^{11}	2^{13}	2^{14}	2^{15}	2^{16}	2^{16}	2^{16}
time (μs)	23.7	27.9	38.2	49.8	73.8	99.3	142	262	575
acceptance (%)	73.3	71.3	67.9	60.9	49.5	40.7	33.4	19.6	10.6

Table 2: acceptance probability and average time for the generation of standard multinormal distributed points

of cones. (Thus FIND() is the most expensive part of the SETUP().) Table 3 shows the situation for the multinormal distribution with density proportional to $f(\mathbf{x}) = \exp(-\sum_{i=1}^{4} i \cdot x_i^2)$, $n = 4$. It demonstrates the effects of continuing barycentric subdivision of the "oldest" edge (see §2.4) on the number of cones, the acceptance probability, the costs for generating a random point proportional to the hat function (i.e. without rejection) and proportional to the given density. Furthermore it shows the total time (in ms) for the setup (i.e. for computing the parameters of the hat function) (in ms) and the time for each cone (in μs) (The increase for large n in the time needed for generating points below the hat is due to effects of memory access time.)

subdivisions	0	1	2	3	4	5	6	7	8	9	10
cones	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
acceptance (%)	26.2	34.1	41.5	48.1	55.3	60.1	64.1	66.6	68.5	69.7	70.5
hat (μs)	24.8	24.8	24.9	25.0	25.2	25.3	25.7	26.1	27.1	27.6	28.4
density (μs)	94.2	73.0	59.9	52.1	45.6	42.9	40.1	39.3	39.5	39.6	40.4
setup (ms)	11.2	22.6	47.2	92.2	182	366	744	1549	3120	6254	12520
setup/cone (μs)	700	706	738	720	713	714	727	756	762	763	764

Table 3: time for computing the hat function for multinormal distribution ($f(\mathbf{x}) = \exp(-\sum_{i=1}^{4} i \cdot x_i^2)$, $n = 4$)

If we do not run FIND() for every cone of the triangulation but use the method described in §3.1 we can reduce the costs for the construction of the hat function. Table 4 gives an idea of this reduction for the multinormal distribution with proportional to $f(\mathbf{x}) = \exp(-\sum_{i=1}^{4} i \cdot x_i^2)$, $n = 4$. It shows the time for constructing the hat function subject to the number of cones for which FIND() is called.

Due to Table 4 the acceptance probability is not very bad, if we run FIND() only for the initial cones. But this is not true in general. It might become extremely poor if the level sets of the density are very "skinny". Table 5 demonstrates the effect on the density proportional to $f(\mathbf{x}) = \exp(-x_1^2 - 10^{-8} x_2^2)$, $n = 2$.

At last table 6 demonstrates that the increase in the time for constructing the hat function for increasing dimension n is mainly due to the increase of

FIND() in subdivision	0	1	2	3	4	5	6
cones (FIND())	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
acceptance (%)	56.4	58.7	60.5	62.1	63.2	63.7	64.1
setup (ms)	66.2	76.8	100.2	141.6	224.7	393.6	744
setup/cone (total) (μs)	65	75	98	138	219	384	727

Table 4: time for computing the hat function for multinormal distribution with "inherited" construction points ($f(\mathbf{x}) = \exp(-\sum_{i=1}^{4} i \cdot x_i^2)$, $n = 4$, 2^{10} cones)

FIND() in subdivision	1	2	3	4	5	6	7	8	9	10
acceptance (%)	0.000 166	0.000 638	0.002 53	0.010 9	0.040 3	0.161	0.635	2.42	7.94	14.82

Table 5: acceptance probability for multinormal distribution with "inherited" construction points ($f(\mathbf{x}) = \exp(-x_1^2 - 10^{-8} x_2^2)$, $n = 2$, 2^{12} cones)

number of cones. Notice that we start with 2^n cones. Furthermore we have to make $(n-1)$ consecutive subdivisions to shorten every edge of a simplex that defines an initial cone. Thus the number of cones increases exponentially.

n	2	3	4	5	6	7	8	9	10
cones	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
acceptance (%)	73.6	70.7	60.1	45.6	31.2	22.3	14.8	9.33	5.77
setup (ms)	69.1	157.8	365.8	830.4	1899	4141	9566	20 609	46 547
setup/cone (μs)	540	616	714	811	927	1011	1170	1250	1421

Table 6: time for computing the hat function for multinormal distribution ($f(\mathbf{x}) = \exp(-\sum_{i=1}^{n} i \cdot x_i^2)$, 5 subdivisions of the initial cones)

If the covariance matrix of the multinormal distribution is not a diagonal matrix and the ratio of the highest and lowest eigenvalue is large, then we cannot use initial cones only and we have to make several subdivisions of the cones. Because of the above considerations the necessary number of cones explodes with increasing n. Thus in this case this method cannot be used for large n. (Suppose we have to shorten every edge of each simplex, then we have $2^3 = 8$ cones if $n = 2$, but we need $2^{19} = 524\,288$ cones for $n = 10$.)

Tests. We ran a χ^2-test with the density proportional to $\exp(-\sum_{i=1}^{3} i\, x_i)$, $n = 3$, to validate the implementation. For all other densities we compared the observed rate of acceptance to the expected acceptance probability.

Comparison with algorithm [LH98]. The code for algorithm [LH98] is much longer (and thus contains more bugs). The setup is much slower and it needs 11 750 μs to generate on mulitnormal distributed random point in dimension 4 (versus 38 μs in table 2 for TDRMV()).

5.3 Rectangular domain

Normal densities restricted to an arbitrary rectangle have a similar performance as the corresponding unrestricted densities. except of the acceptance probability which is worse since the domain of the hat h is a superset of the domain of density f.

5.4 Quality

The quality of non-uniform random number generators using transformation techniques is an open problem even for the univariate case (see e.g. [Hör94] for a first approach). It depends on the underlying uniform random number generators. The situation is more serious in the multivariate case. Notice that this new algorithm requires more than $n+2$ uniform random numbers for every random point. We cannot give an answer to this problem here, but it should be clear that e.g. RANDU (formerly part of IBM's Scientific Subroutine Package, and now famous for its devasting defect in three dimensions: its consecuting points (x_i, x_{i+1}, x_{i+2}) lie in just fifteen parallel planes; see e.g. [LW97]) may result in a generator of poor quality.

5.5 Some Examples

We have tested our algorithm in dimensions $n = 2$ to $n = 8$ with densities proportional to

$$f_1(\mathbf{x}) = \exp(-\sum a_i\, x_i^2)$$

$$f_2(\mathbf{x}) = \exp(-\sum a_i\, |x_i|)$$

$$f_3(\mathbf{x}) = \max(0, \prod(1 - a_i\, x_i^2))$$

$$f_4(\mathbf{x}) = \max(0, \prod(1 - |x_i|^{a_i})$$

$$f_5(\mathbf{x}) = \max(0, \sum(1 - a_i\, x_i^2))$$

where $a_i > 0$. The domain was $\mathbb{R}^n$ and some rectangles. We also used densities proportional to $f_i(U\,\mathbf{x} + \mathbf{b})$, where U is an orthonormal transformation and $\mathbf{b}$ a vector, to test distributions with non-diagonal correlation matrix and arbitrary mode.

The algorithm works well for densities f_3, f_4 and f_5 both with $D = \mathbb{R}^n$ and D being a rectangle enclosing the support of f_i. Although some of these densities are not $\mathcal{C}^1$, the FIND() routine works. Problems arise if the level sets of the density have "corners", i.e. the $\mathbf{g}$ is unstable when we vary the touching

p a little bit. Then there are somes (that contains these "corners") with huge volumeH_C and further triangulation is necessary. If dimension is high ($n \gtrsim 5$) too many cones might be necessary. The optimization algorithm for finding the mode fails if we use a starting point outside the support of f_5.

The code has some parameters for adjusting the algorithm to the given density. For example, it requires some testing to get the optimal number of cones and the optimal level of subdivisions for calling FIND().

5.6 Résumé

The presented algorithm is a suitable method for sampling from log-concave (and T-concave) distributions. The algorithm works well for all tested log-concave densities if dimension is low ($n \lesssim 5$) or if correlation is not too high. Restrictions of these densities to compact polytopes are possible. The setup time is small for small dimension but increases exponentially in n. The speed for generating random points is quite fast even for $n \geq 6$. Due to the large amount of cones for high dimension the program requires a lot of computer memory (typically 2–10 MB).

Although the developed algorithm is not a real black box method it is easily adjustable for a large class of log-concave densities. Examples for which the algorithm works are the multivariate normal distribution and the multivariate student distribution (with transformation $T(x) = -x^c$) with arbitrary mean vector and variance matrix conditioned to an arbitrary compact polytope. However for higher dimensions the ratio of highest and lowest eigenvalue of the covariance matrix should not be "too big".

Acknowledgments

The author wishes to note his appreciation for help rendered by Jörg Lenneis. He has given lots of hints for the implementation of the algorithm. The author also thanks Gerhard Derflinger and Wolfgang Hörmann for helpful conversations and their interest in his work.

References

[AD82] J.H. Ahrens and U. Dieter. Generating gamma variates by a modified rejection technique. *Comm. ACM*, 25(1):47–54, January 1982.

[Ahr95] J. H. Ahrens. A one-table method for sampling from continuous and discrete distributions. *Computing*, 54:127–146, 1995.

[BM58] G.E.P. Box and M.E. Muller. A note on the generation of random normal deviates. *Annals of Mathem. Stat.*, 29(2):610–611, 1958.

[BP66] M. Bell and M. C. Pike. Remark on algorithm 178. *Comm. ACM*, 9(9):684–686, September 1966.

[Dag88] J. Dagpunar. *Principles of Random Variate Generation*. Clarendon Press, Oxford, 1988.

[Dev86] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New-York, 1986.

[Dev97] L. Devroye. Random variate generation for multivariate densities. *ACM TOMACS*, 7(4):447–477, October 1997.

[DH94] G. Derflinger and W. Hörmann. The optimal selection of hat functions for rejection algorithms. manuscript, private communication, 1994.

[ES97] M. Evans and T. Swartz. Random variable generation using concavity properties of transformed densities. *J. of Comp. and Graph. Stat.*, 1997. to appear.

[FMM77] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer methods for mathematical computations*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, NJ, 7 edition, 1977.

[GR65] I. S. Gradshteyn and I. M. Ryzhnik. *Table of Integrals*. Academic Press, 5th edition, 1965.

[Grü67] B. Grünbaum. *Convex Poytopes*. Interscience, 1967.

[GW92] W. R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Appl. Statistics*, 41:337–348, 1992.

[HD94] W. Hörmann and G. Derflinger. Universal generators for correlation induction. In R. Dutter and W. Grossmann, editors, *Compstat, Proceedings in Computational Statistics*, pages 52–57, Heidelberg, 1994. Physica-Verlag.

[HJ61] R. Hooke and T. A. Jeeves. "Direct search" solution of numerical and statistical problems. *J. ACM*, 8(2):212–229, 1961.

[Hör94] W. Hörmann. The quality of non-uniform random numbers. In H. Dyckhoff et al., editors, *Operations Research Proceedings 1993*, pages 329–335, Berlin, 1994. Springer Verlag.

[Hör95a] W. Hörmann. A rejection technique for sampling from T-concave distributions. *ACM Trans. Math. Software*, 21(2):182–193, 1995.

[Hör95b] W. Hörmann. A universal generator for bivariate log-concave distributions. Preprint, 1995.

[JKB95] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. Wiley–Interscience, New York, 2nd edition, 1995.

[Joh87] M. E. Johnson. *Multivariate Statistical Simulation.* John Wiley & Sons, New York, 1987.

[Kau63] A. F. Kaupe Jr. Algorithm 178: Direct search. *Comm. ACM*, 6(6):313–314, June 1963.

[Law91] J. Lawrence. Polytope volume computation. *Math. Comput.*, 57(195):259–271, 1991.

[Ley98] J. Leydold. *TDRMV — Generating multivariate log-concave densities with transformed density rejection.* Institut für Statistik, Wirtschaftsuniversität Wien, 1998. code available at `ftp://statistik.wu-wien.ac.at/src/tdrmv/`.

[LH98] J. Leydold and W. Hörmann. A sweep plane algorithm for generating random tuples. *Math. Comp.*, 1998. to appear.

[LW97] H. Leeb and S. Wegenkittl. Inversive and linear congruential pseudorandom number generators in empirical tests. *ACM TOMACS*, 7(2):272–286, 1997.

[Pré73] A. Prékopa. On logarithmic concave measures and functions. *Acta Sci. Math. Hungarica*, 34:335–343, 1973.

[Rao84] S. S. Rao. *Optimization. Theory and Applications.* Wiley Eastern Ltd., New Delhi, 2nd edition, 1984.

[SV87] S. Stefănescu and I. Văduva. On computer generation of random vectors by transformations of uniformly distributed vectors. *Computing*, 39:141–153, 1987.

[Tod76] M. J. Todd. *The Computation of Fixed Points and Applications*, volume 124 of *Lecture Notes in Economics and Mathematical Systems.* Springer, Berlin, 1976.

[Tod78] M. J. Todd. Improving the convergence of fixed-point algorithms. *Mathematical Programming Study*, 7:151–169, 1978.

[WGS91] J. C. Wakefield, A. E. Gelfand, and A. F. M. Smith. Efficient generation of random variates via the ratio-of-uniforms method. *Statist. Comput.*, 1:129–133, 1991.

[Zie95] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1995.