An Automatic Generator for Bivariate Log-Concave Distributions

Wolfgang Hörmann
Bogazici University, Istanbul

Different automatic (also called universal or black-box) methods have been suggested to sample from univariate log-concave distributions. Our new automatic algorithm for bivariate log-concave distributions is based on the method of transformed density rejection. In order to construct a hat function for a rejection algorithm the bivariate density is transformed by the logarithm into a concave function. Then it is possible to construct a dominating function by taking the minimum of several tangent planes, which are by exponentiation transformed back into the original scale. The choice of the points of contact is automated using adaptive rejection sampling. This means that points that are rejected by the rejection algorithm can be used as additional points of contact. The paper describes the details how this main idea can be used to construct Algorithm ALC2D that can generate random pairs from all bivariate log-concave distributions with known domain, computable density and computable partial derivatives.

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

General Terms: Algorithms

Additional Key Words and Phrases: Rejection method, bivariate log-concave distributions, automatic generator, universal generator

1. INTRODUCTION

For the univariate case there is a large literature on generation methods for standard distributions (see eg. [Devroye 1986] and [Dagpunar 1988]) and in the last years some papers appeared on automatic (also called universal or black-box) methods (see [Devroye 1986] chapter VII, [Gilks and Wild 1992], [Ahrens 1995], [Hörmann 1995] and [Hörmann and Deflinger 1994]); these are algorithms that can generate

This work was partially supported by the Austrian Research Council (FWF), project J01555-mat and by the Austrian Academy of Science, APART-scholarship.
Address: Department of Industrial Engineering, 80815 Bebek-Istanbul, Turkey.
e-mail: hormannw@boun.edu.tr

random variates from a large family of distributions 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 is only the generation of the multinormal and of the Wishart distribution (see eg. [Devroye 1986] and [Dagpunar 1988]). A different approach – especially considered by researchers interested in simulation – aims to develop new, easy to generate, classes of bivariate distributions; it is only necessary (and possible) to specify the marginal distribution and the degree of dependence measured by some correlation coefficient (see the monograph [Johnson 1987] or [Devroye 1986] chapter XI). This idea seems to be attractive for many simulation practitioners interested in bivariate distributions but it is no help if variates from a bivariate distribution with given density should be generated.

For this task one help available in books on simulation is the description of the conditional distribution method, which requires the knowledge of and the ability to sample from the marginal and the conditional distributions. Another general hint says that the decomposition and rejection method can be applied to multivariate distributions as well. One majorizing function (also called hat-function) suggested for the multivariate rejection method is the product of the marginal densities (in [Dagpunar 1988]) but it is not clear at all how to obtain the necessary multiplicative constant for the hat-function. Another possibility is the multivariate extension of the ratio of uniforms method (for the univariate method see [Devroye 1986] and references given there) as described in the – unfortunately little known – paper [Stefănescu and Văduva 1987] and "rediscovered" in [Wakefield et al. 1991]. The multivariate ratio of uniforms 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, one disadvantage of that method which is mentioned in [Wakefield et al. 1991]. 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 [Stefănescu and Văduva 1987] and [Wakefield et al. 1991] and seems to be difficult for most distributions.

In summer 1995, when we finished the first version of this paper and coded a first prototype of the new algorithm, this was – up to our knowledge – all one could find in the literature about the generation of variates from a bivariate distribution with given density function. No automatic algorithms were known. In [Devroye 1986] p. 557 it is even stressed that no general inequalities for multivariate log-concave densities are available, a fact which makes the design of black-box algorithms, similar to those developed in [Devroye 1986] for the univariate case, impossible.

Since 1995 research on automatic generators for multivariate distributions has been intensivated by L. Devroye in Montreal and in Vienna, mainly by J. Leydold, as the author was loaded with other work. The mathematics necessary for a sweep-plane algorithm to generate uniform random variates over simple polytopes in high dimensions was collected in [Leydold and Hörmann 1998]. Unfortunately the sketched algorithm for multivariate log-concave distributions turned out to be complicated and slow. Therefore J. Leydold (cf. [Leydold 1998]) developed a different method that is in practice applicable to log-concave distributions up to dimension 10, if the correlation is not too big. In the same time L. Devroye ([Devroye 1997])

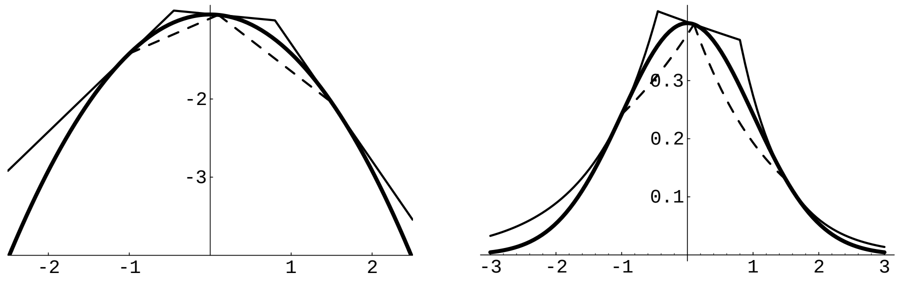

Fig. 1. Normal distribution with points of contact -1, 0.1 and 1.5
left hand side: logarithmic scale; $g(x)$ ...thick line; $l(x)$ ...thin line; squeeze ...dashed line
right hand side: original scale; $f(x)$ ...thick line; $h(x)$ ...thin line; squeeze ...dashed line

considers algorithms for a large class of multivariate distributions called orthounimodal. His algorithms also work up to dimension 10. In [Devroye 1997] it is also stated that log-concave densities require a separate study. We are convinced that the two-dimensional case is of great practical importance, but the two methods for log-concave densities suggested in [Leydold and Hörmann 1998] and [Leydold 1998] are designed to work in higher dimensions and have clear disadvantages when used for two-dimensional distributions.

Therefore we present a new generator for bivariate log-concave distributions. Given a computable, log-concave density $f(x, y)$ and its partial derivatives and the domain of the distribution the algorithm can generate random variates (X, Y) with the desired distribution almost as fast as independent normal pairs can be generated by the Box-Muller method. It is a generalisation of the univariate universal algorithm for log-concave distributions presented in a first form in [Devroye 1986] chapter VII.2.4 and with a different set-up in [Gilks and Wild 1992]. As the new algorithm uses the property of the R^2 that every convex polygon can be triangulated without problems the direct generalisation of this algorithm to dimensions higher than two is of no practical relevance compared with the algorithm developed in [Leydold 1998]. It is possible to generalise our new algorithm to two-dimensional T-concave distributions (for T-concave see [Hörmann 1995]) but for the sake of simplicity we restrict our attention in this paper to the log-concave bivariate case. Section 2 explains the idea of transformed density rejection for one and two dimensions, Section 3 provides the details of the algorithm and Section 4 reports the computational experience we made with the new algorithm.

2. TRANSFORMED DENSITY REJECTION

To design a universal algorithm utilising the rejection method it is necessary to find an automatic way to construct a hat function for a given density. Transformed density rejection was introduced under a different name in [Gilks and Wild 1992], generalised in [Hörmann 1995] and is also considered with further generalisations in [Evans and Swartz 1998]. It is based on the idea that the density f is transformed by a monotone transformation T in such a way that $g(x) = T(f(x))$ is concave. Then it is simple to construct a hat $l(x)$ for $g(x)$ as the minimum of N tangents touching $g(x)$ in N points. As g is concave it is clear that we have $g(x) \leq l(x)$

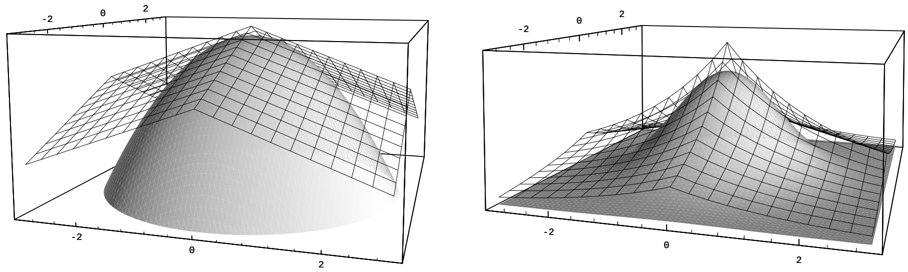

Fig. 2. Bivariate standard normal distribution; four points of contact $(\pm 0.5, \pm 0.5)$
left hand side: logarithmic scale; $g(x,y)$ ...solid surface; $l(x,y)$ ...grid
right hand side: original scale; $f(x,y)$ ...solid surface; $h(x,y)$ ...grid

for all x. Transforming $l(x)$ back into the original scale we get $h(x) = T^{-1}(l(x))$ with $f(x) \leq h(x)$ as majorizing function or hat for f. As we are only interested in log-concave densities we take $T(x) = \log(x)$ in the sequel. Then $h(x)$ consists of exponential pieces. We get a simple to calculate lower bound for the density – often called squeeze in random variate generation – by linear interpolation of the points of contact as shown in Figure 1. This squeeze reduces the expected number of evaluations of f.

The main idea of this paper is to extend transformed density rejection to the bivariate case. First we present the basic form of the bivariate rejection method.

Algorithm 1. *Rejection*
Set-up: Construct a hat-function $h(x,y)$.
Repeat
  Generate a random pair (X,Y) *with density proportional to* $h(x,y)$
  Generate a uniform random number V.
Until $Vh(X,Y) \leq f(X,Y)$
Return (X,Y).

To manage the set-up of the above algorithm we transform the bivariate density $f(x,y)$ into $g(x,y) = \log(f(x,y))$ which is concave; then we construct tangent planes in several points (called points of contact or design points in the sequel) and take the pointwise minimum of these planes as $l(x,y)$; $h(x,y) = \exp(l(x,y))$ is then a hat function for the bivariate density f (see Figure 2).

The idea of the squeezes could be generalised to two dimensions as well; in the logarithmic scale the plane defined by three points of contact is a lower bound for g as long as (x,y) lies inside the triangular defined by these three points. Unfortunately the search for the triangular, where the squeeze is maximal, is so slow and complicated that we decided not to include squeezes in the new algorithm.

Although the main idea of bivariate transformed density rejection is simple and can be understood by looking at Figure 2 it is not simple to collect all necessary details for the rejection algorithm.

Different to the univariate case it is not so easy to generate variates from the bivariate distribution with density proportional to the hat $h(x,y)$. As h is the

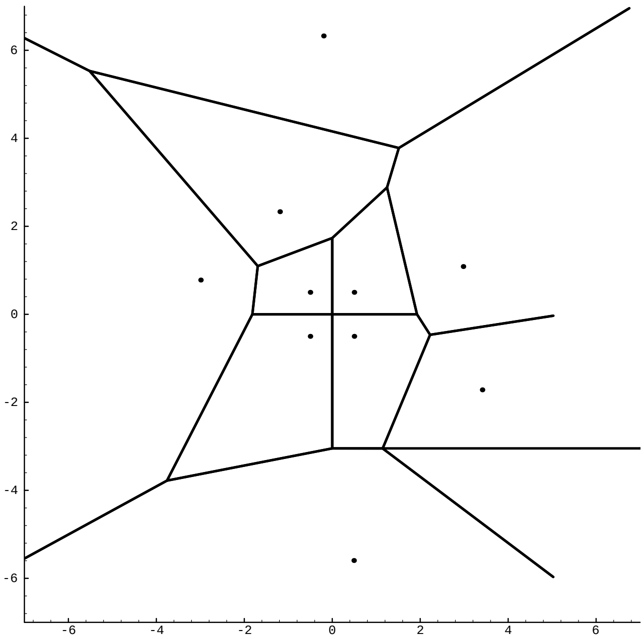

Fig. 3. Ten points of contact and the resulting partition of the plane into ten polygons for the bivariate standard normal distribution

pointwise minimum of N tangent planes transformed back by $\exp(x)$ it is first necessary to compute the domains of the different parts of $h(x, y)$, which are grouped around their respective points of contact. We find that convex (possibly open) polygon for a fixed point of contact (x_0, y_0) in two steps: First it is necessary to compute the $N-1$ intersection lines of the tangent plane touching in (x_0, y_0) with all other tangent planes. We store the projections p_i of these lines to the (x, y)-plane. Then the polygon is the intersection of all half-planes containing (x_0, y_0) bordered by the p_i. For the standard bivariate normal distribution and the points of contact $(\pm 0.5| \pm 0.5)$ (this is the situation shown in Figure 2) the four (open) polygons are simply the four quadrants of the (x, y)-plane. Figure 3 shows the result after adding 6 design points randomly.

After computing the polygon it is translated such that the corner with the maximal value for $h(x, y)$ is transformed into the origin. Then it is rotated such that the tangential plane $g(x, y) = t_1 x + t_2 y + t_3$ can be written as $ax + s$ which is established if the negative gradient of g, $-\nabla g = (-t_1, -t_2)$ is rotated into $(\sqrt{t_1^2 + t_2^2}, 0)$. a is obviously the negative length of the gradient $a = -\sqrt{t_1^2 + t_2^2}$, s the value of the translated tangential plane in the origin. Elementary algebra shows that we have to use the rotation $\begin{pmatrix} r_1 & -r_2 \\ r_2 & r_1 \end{pmatrix}$ with

$$r_1 = \frac{-t_1}{\sqrt{t_1^2 + t_2^2}}, \qquad r_2 = \frac{t_2}{\sqrt{t_1^2 + t_2^2}}, \qquad a = -\sqrt{t_1^2 + t_2^2} \tag{1}$$

or, if $a = 0$, we choose $r_1 = 1$, $r_2 = 0$.

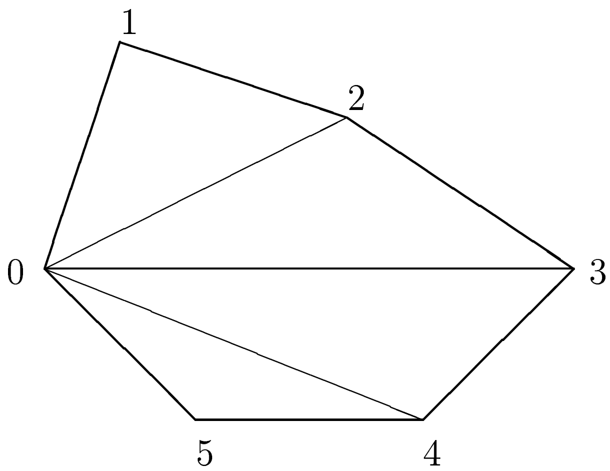

Fig. 4. Triangulation of a polygon; Corner 0 has maximal hat-value

Starting with the corner in the origin, the translated, rotated polygon is decomposed into triangles. (See Figure 4 for an example.)

We can see that this triangulation is a simple and fast procedure in two dimensions and it can be shown that the total number of triangles grows linearly with the number of design points (cf. Section 4), which is no longer true for dimensions larger than two.

As shown in Figure 5, each triangle is decomposed into two generator regions (g_1 and g_2) by the line p parallel to the y-axis through that vertex of the transformed triangle that is closer to the y axis. b denotes the distance between that line and the y-axis. For the generation we also need the slopes k_0 and k_1 of the lines connecting the origin with the two vertices of the transformed triangle. The hat over the generator region is $h(x,y) = \exp(ax+s)$. So we easily find the marginal density $f_Y(x) = (k_1 - k_0)xe^{ax+s}$. The volume below the hat is

$$\text{vol} = \int_0^b f_Y(x)dx = \frac{e^s(k_1-k_0)}{a}(e^{ab}(b-1/a)+1/a) \tag{2}$$

The part of the triangle on the right hand side of line p is the second generator region (cf. Figure 5). In the closed case the constants necessary for the second generator region are computed analogously to those of the first generator region, only the third vertex is translated into the origin and the rotation is into the opposite direction. This also implies that we have to replace a by $-a$.

To generate random pairs from the hat over a bounded generator region we must be able to sample from the marginal distribution $x\exp(ax)$ in the interval $(0,b)$. This is no problem for the special case $a=0$. For $a \neq 0$ it is best to generate X in the interval $(0,|ab|)$ with density proportional to $x\exp(x)$ or $x\exp(-x)$ depending

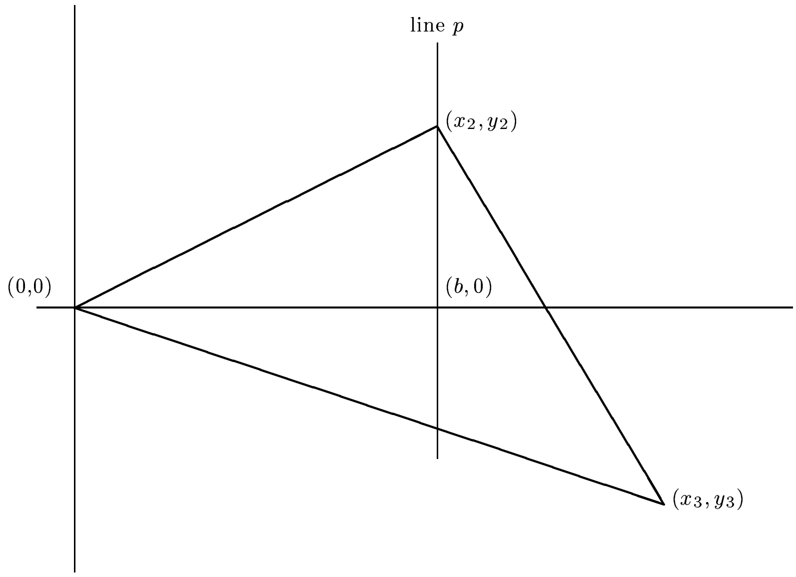

Fig. 5. Bounded case: g_1 ... part of the triangle left of p; g_2 ... part of the triangle right of p

on the sign of a and to return $X/|a|$. As the values of a and b vary at every call it is important to find a generator with a fast set-up. After some experimentation we used a rejection algorithm with a fixed linear-constant-exponential hat for $x\exp(-x)$, and an exponential hat touching at $x_0 = 0.65|ab|$ for $x\exp(x)$. As the speed of this marginal generator has a considerable influence on the speed of the whole algorithm we use squeezes to accelerate the generation.

For an open polygon after triangulation there remains an unbounded region, either a simple angle or an open polygon consisting of two vertices and two lines towards infinity. From this region we can cut off a bounded generator region (cf. Figure 6), that is the same as the first generator region of a bounded triangle. After cutting off the bounded region the rest, which is the second generator region, can be described as a single angle and a parallel strip. (The parallel strip has width 0 for the case that the whole open polygon is only a single angle.) The second generator region of the unbounded case is rotated in the same way as the first region but the second vertex is translated into the origin. k_0 and k_1 are the slopes of the two lines l_1 and l_2 towards infinity and b is the y-coordinate of the intersection of the second line with the y-axis (cf. Figure 6).

For the generation of the marginal density in x-direction for an open generator region we consider the region as divided into a parallel strip and an angle by the line p_2 through the point $(0, b)$ with slope k_2. Then it is easy to see that the marginal density is the mixture of an exponential density (e^{-ax}) (coming from the parallel strip) and a gamma(2) density (xe^{-ax}) (coming from the angle), both with scale

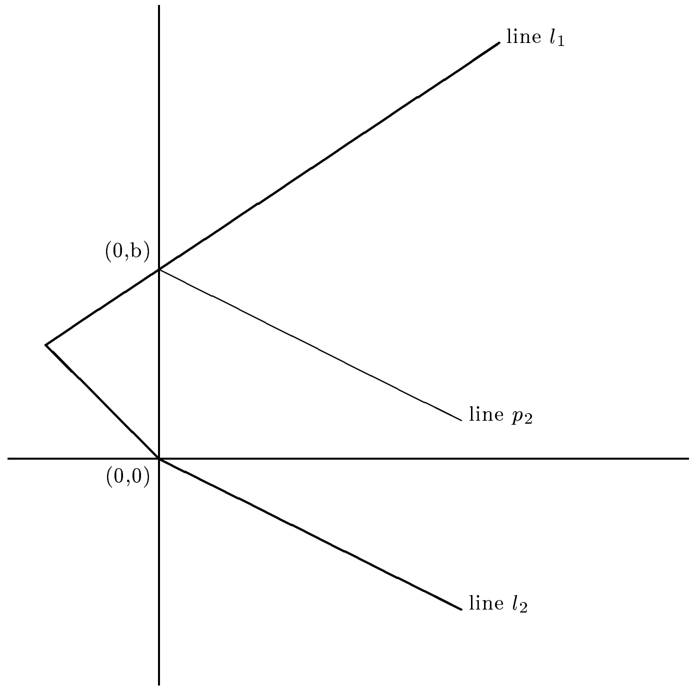

Fig. 6. Unbounded case: g_1 ...triangle on the left hand side of the y-axis; g_2 ...unbounded region bordered by l_1, the y-axis and l_2

parameter a. The weights of the mixture depend on the volume below these two parts of the region and decomposition can be used to generate the marginal density. For the volumes below the hat we get:

$$\mathrm{vol}(angle) = \int_0^\infty f_y(x)dx = e^s(k_1 - k_0)/a^2 \tag{3}$$

$$\mathrm{vol}(parallelstrip) = e^s b/(-a)\mathrm{vol} = ((k_1 - k_0)/a - b)e^s/a \tag{4}$$

Now we have explained how to divide the domain of the distribution into generator regions, how to compute the volume below the hat for the different generator regions and how to sample from the marginal distribution along the x-axis. To sample from the hat over a generator region we have to add the remark that the conditional distribution of Y for given x is uniform. (Note that even for an unbounded polygon for any given x the domain of Y is bounded, if the volume below the hat is bounded.) Thus it is easy to generate random variates with density proportional to $h(x, y)$ over any of the regions by using the conditional distribution method. To decide between the different regions of the hat the volume between $h(x, y)$ and the (x, y)-plane for each region must be computed and the decomposition method can be utilised to decide, which region should be used to generate the random variate.

The second problem, that must be overcome, is the suitable choice of the points of contact. In one dimension we intended (see [Hörmann 1995] and [Hörmann and Deflinger 1994]) to optimise the choice of the points of contact which made it also possible to show that the execution time of the algorithm is uniformly bounded for a family of T-concave distributions. In two dimensions optimisation seems to be very difficult. Instead we use the important idea of adaptive rejection sampling introduced in [Gilks and Wild 1992]. Adapted to our situation it works in the following way: First take some points of contact, (called starting values) which must have only the property that the volume below the hat $h(x,y)$ is bounded. Then start the generation of random variates with this hat until one pair (x,y) is rejected; take that pair as an additional point of contact, construct the new hat and restart generation of the variates; every rejected point is taken as additional point of contact until a certain stopping criterion is fulfilled, e.g. the maximal number N of design points or the aimed acceptance probability is reached. Using this rule the points of contact are chosen by a stochastic algorithm and it is clear that the bivariate density of the distribution of the next point of contact is proportional to $h(x,y) - f(x,y)$. Thus with N tending towards infinity the hat-function converges against the density with probability 1. It is not difficult to show the following:

THEOREM 1. *For a two times differentiable log-concave density function f, and N design-points placed on a regular grid, the expected volume below the hat is*

$$\alpha = 1 + O(1/N).$$

PROOF. Without loss of generality we may restrict our considerations to densities with bounded domain. As h and f are both two times differentiable functions with the same first-order Taylor-expansion in the design point, we have (for r denoting the distance from the design point) $|h - f| = O(r^2)$ around each design-point. If we have N design points on a regular grid the average radius is $r = O(1/\sqrt{N})$, which implies that the average distance of h and $f = O(1/N)$. As we are assuming a bounded domain we get: $\int_{\text{domain}} |h(x,y) - f(x,y)| = O(1/N)$ $\square$

Using the same idea it is no problem to prove:
For arbitrary dimension $d \geq 1$, $\quad \alpha = 1 + O(N^{-2/d})$.

Using adaptive rejection sampling the design points do not form a regular grid. Although we do not yet have a strict mathematical proof for it, it seems to be obvious, and is very strongly supported by our computational experience, that the results for adaptive rejection sampling (and our new algorithm) are better than using a regular grid of design points and follow the same $\alpha = 1 + O(1/N)$ law.

It is necessary to find starting values such that the volume below the hat is bounded before we can utilise the idea of adaptive rejection sampling. For densities with bounded domain this is a simple task: Just take one or more points not too far away from the mode. For distributions with unbounded domain the problem can be more difficult since the volume below the hat might be unbounded. After trying several ideas we arrived at the following suggestion: Define a bounded auxiliary domain, which must contain the mode and should contain the main region of the distribution. Then use one starting value, if possible close to the mode, and adaptive rejection sampling to find design points for a moderately good-fitting hat for the density restricted to the auxiliary domain. These design points are then used as starting values for constructing a hat for the density over the unbounded domain. In Section 4 we report the remarkably good results that we obtained with this method for choosing the starting points.

3. THE ALGORITHM IN DETAIL

In this section we try to give the main design of Algorithm ALC2D (automatic log-concave 2-dimensional) that can sample from arbitrary bivariate log-concave distributions given that all information described as input of subprogram Setup below is available. As our C-program (which is available from the author on e-mail request) has about 1500 statements a description of all details is out of question. We purposely do not discuss the question of data structures and memory management here. Our solution of these problems can be studied in the C-code. We tried to waste not too much memory and to guarantee fast access using dynamic arrays, but we do not claim that our implementation is optimal or very elegant in that respect.

To use Algorithm ALC2D in the below formulation it is necessary for the user to call set-up once for initialisation; then sample2d can be used to generate variates from the desired bivariate distribution, sample2d automatically adds new points of contact and calls the add-design-points procedure when points are rejected as long as the maximal number of points of contact N or the aimed acceptance probability p_a is not exceeded.

First it is necessary to define a data-structure that contains all information associated with a single point of contact and its polygon, and a second data-structure that stores all information associated with one generator region. The details are given in Table 1.

Subprogram: **Set-up**
Input:

—The logarithm of the density $\log(f(x,y))$, which must be a concave function on the entire domain; (it is not necessary that f integrates to 1);

—the partial derivatives of $\log(f(x,y))$;

—the domain of the distribution which must be a convex region bordered by a (possibly open) polygon or the whole plane,

—one or several points of contact to start with; they should be not too far away from the mode;

—for distributions with unbounded domain a bounded auxiliary domain; the mode must be in the interior of that domain. The value of the density on the border of the domain should be small compared with the maximal value of the density.

—the maximal number of design points N that can be used

—optional: the aimed acceptance probability p_a and the (approximate) volume below the density function v_f.

Set-up computes the hat and all constants for the rejection algorithm.

Table 1. Data Structures

		Polygon
$p.n_c$	integer	current number of corners of polygon+unbounded edges
$p.o$	integer	0 for closed, 1 for open polygon
$p.d$	2 floats	design-point
$p.t_0, p.t_1, p.t_2$	3 floats	tangential plane with equation: $z = p.t_0 + p.t_1 x + p.t_2 y$
$p.c_0$	2 floats	
$\vdots$	2 floats	vertices of the polygon in consecutive order
$p.c_{p.n_c-1}$	2 floats	
$p.sc$	2 floats	vertex of the polygon with maximal hat-value
$p.r =$	2 floats	rotation that rotates the gradient of the tangential plane in x-direction
$(p.r_1, p.r_2)$		the matrix of the rotation is $\begin{pmatrix} r_1 & -r_2 \\ r_2 & r_1 \end{pmatrix}$
$p.s$, $p.a$	2 floats	constants for the hat
		Generator region
$g.o$	integer	0 closed, 1 open generator region
$g.p$	2 floats	original coordinates of the point that was translated into origin
$g.r$	2 floats	rotation
$g.b$	float	broadness of generator region
$g.k_0, g.k_1$	2 floats	slopes of the two border-lines
$g.s, g.a$	2 floats	$h(x, y) = \exp(g.a * x + g.s)$ is the hat over the rotated generator region
$g.vol$	float	volume below the hat for this region

Allocate the necessary memory.
Set all polygons equal to the domain of the distribution.
Store the starting points as design points for the first polygons.
`Call` **add-design-points** to compute all constants necessary for generation.
`If` the volume below the hat is unbounded,
  `Call` **set-up** using the auxiliary domain as the domain of the distribution.
  `If` volume below the hat over the auxiliary domain is unbounded,
    `Exit` with error message.
  `Repeat:`
    `If` the maximal number N of design points is reached,
      `Exit` with error message.
    `Call` **sample2d** for the distribution over the auxiliary domain.
    (Not because the random pair is needed but because the hat is improved.)
    `Call` **add-design-points** with the original domain, using the
    starting points and all design-points found during the calls to sample2d.
  `Until` the volume below the hat over the original domain is bounded.

Subprogram: **Add-design-points**
Input: Points of contact and the domain of the distribution
Decomposes the domain into generator regions and computes all constants.

Compute for all new points of contact the tangential planes.
Initialise the polygons for these points with the domain of the distribution.
`For` all new design points $p_i.d$:
  `For` all other design points $p_j.d$ with $j \neq i$:
    Compute the projection l of the intersection of the two tangential planes.
    l has the equation: $0 = t_{i,0} - t_{j,0} + x(t_{i,1} - t_{j,1}) + y(t_{i,2} - t_{j,2})$.
    `If` the two tangential planes are almost identical,
      remove the new design point $p_i.d$.
    `Else`
      `Call` **polyupdate** with (p_i, l),
      `Call` **polyupdate** with (p_j, l),
      (to update p_i and p_j).
`For` all polygons p_k:
  `Call` **triangulate**.
Number all generation regions in consecutive order;
compute the cumulated volumes.
Prepare a guide table for the cumulated volumes.
This speeds up the generation of the discrete variate.

Subprogram: **Polyupdate**
Input: A polygon and the line l defined by: $0 = l_0 + l_1 x + l_2 y$;
Computes the polygon updated by l such that the design point remains inside.

`Do` one of the three cases:
`Case1:` The polygon has been entirely empty up to now:
  Store the line, using the space of the corners c_0 and c_1.
`Case2:` The polygon consists of only one line up to now:
  Store the intersection with the new line as corner c_1,
  store the direction vectors of the two lines, as c_0 and c_2.
  Change the signs of c_0 and c_2 such that the point of contact
  is inside the angle with less than 180^0.
  The necessary details for providing for the case that the second line is parallel to the first one are lengthy and therefore omitted here.
`Case3:` The polygon has at least one vertex:
  Compute for all vertices, if the vertex remains or is cut off by the line.
  (Also for the two vectors c_0 and c_{n_c-1} of an open polygon.)
  `If` there are vertices that should be cut off,
    Compute the two intersection points between the line and the polygon.
    Dismiss the cut off vertices.
    Instead of these vertices insert the new intersection points.
    (As we interpret the vectors of the polygon as intersection points as well, there are always two intersection-points that must be inserted.)
    `If` the polygon was open before and c_0 and c_{n-1} were cut off,
      $p.o \leftarrow 0$, to indicate, that the polygon is closed now.

Subprogram: **Triangulate**
Input: A polygon
Decomposes the polygon into triangles and generator regions.

Search that corner of the polygon with the maximal value of the tangential plane.
`If` two corners have the same value,
    take the corner with the minimal x-value.
Store the coordinates of the corner in sc,
store the maximal value of the tangential plane as s.
Using formula (1) compute and store the rotation $r = (r_1, r_2)$ and a.
Compute the coodinates $\tilde{c}_i$ of all translated and rotated corners
(using sc and $r = (r_1, r_2)$.)
$\tilde{c}_0 = (0, 0)$ *is the point with the maximal value of the tangential plane.*
`For` all i from 1 to $n_c - 2$:
    `Call` **compgen** with the two corners $\tilde{c}_i$ and $\tilde{c}_{i+1}$.
    This triangulates the translated rotated polygon from the point $\tilde{c}_0 = (0, 0)$.
`If` the polygon is open,
    `call` **compgeno**.
    Computes the two generator regions for the open part of the polygon.

Subprogram: **Compgen**
Input: A polygon and one of its translated rotated triangles ((0,0), v_1, v_2).
Decomposes the triangle into two generation regions (g_1 and g_2)
and computes all constants necessary to generate variates from these regions.

`If` $v_1(x) > v_2(x)$,
    swap v_1 and v_2.
Find v_2^o, the original coordinates of v_2.
`If` $v_1(x) > 0$,
    The following steps compute and store the constants for g_1.
    $g_1.o \leftarrow 0$; $g_1.b \leftarrow v_1(x)$; $g_1.s \leftarrow p.s$; $g_1.r \leftarrow p.r$;
    $g_1.k_0 \leftarrow v_1(y)/v_1(x)$; $g_1.k_1 \leftarrow v_2(y)/v_2(x)$;
    $g_1.p \leftarrow p.sc$; $g_1.a \leftarrow p.a$;
    use formula (2) to compute $g_1.vol$.
`Else` $g_1.vol \leftarrow 0$.
The following steps compute and store the constants for g_2:
$g_2.b \leftarrow v_2(x) - v_1(x)$;
`If` $g_2.b > 0$,
    $g_2.o \leftarrow 0$; $g_2.s \leftarrow p.s + p.a * v_2(x)$;
    $g_2.r_1 \leftarrow -p.r_1$; $g_2.r_2 \leftarrow -p.r_2$;
    $g_2.k_0 \leftarrow (v_2(y) - v_1(y))/g_2.b$; $g_2.k_1 \leftarrow v_2(y)/v_1(x)$;
    $g_2.a \leftarrow -p.a$; $g_2.p \leftarrow v_2^o$;
    use formula (2) to compute $g_2.vol$.
`Else` $g_2.vol \leftarrow 0$.

Subprogram: **Compgeno**
Input: An open polygon
Decomposes the open part of the polygon into two generation regions (g_1 and g_2) and computes all constants necessary to generate variates from these regions.

Compute the value of the tangential plane in the two corners.
Store the larger value as $g_1.s$ and its corner as v_0^o,
store the smaller value as $g_2.s$ and its corner as v_1^o.
Store $g_1.r = p.r$.
Store the translated rotated coordinates of v_1^o as v_1,
store the coordinates of the rotated vector pointing from v_0^o as v_2,
store the coordinates of the rotated vector pointing from v_1^o as v_3.
In the case of a simple angle v_0^o and v_1^o are identical.
Compute and store the missing constants for g_1 analogous to in **compgen**.
The following steps compute and store the constants for unbounded region g_2:
`If` the polygon has only one true corner,
    $g_2.b \leftarrow 0$.
`Else If` $v_1(x) = 0$,
    $g_2.b \leftarrow -v_1(y)$,
`Else`
    $g_2.b \leftarrow g_1.b * g_1.k_1 - g_1.k_0$.
$g_2.o \leftarrow 1$; (*open generator region*)
$g_2.a \leftarrow p.a$;
$g_2.r \leftarrow p.r$;
`If` $v_2(x) > 0$,
    $g_2.k_1 \leftarrow v_2(y)/v_2(x)$;
`Else`
    `exit` with error message: "Volume below hat unbounded!"
`If` $v_3(x) > 0$,
    $g_2.k_0 \leftarrow v_3(y)/v_3(x)$;
`Else`
    `exit` with error message: "Volume below hat unbounded!"
$g_2.p \leftarrow v_1^o$.
Use formulas (3) and (4) to compute $g_2.vol$.

Subprogram: **Sample2d**
Input: All information about polygons and generator regions
The algorithm uses the constants stored in the generator regions, to generate a sample (X, Y) of the distribution with density proportional to $f(x, y)$.

`Repeat:`
    Generate a random integer J with probabilities proportional to $g.vol$.
    J is the number of the generator region, in which the pair is generated.
    Use the guide-table method to accelerate this generation.
    `If` the generator region with index J is bounded,
        Generate a random sample X from the marginal distribution.
        The density of X is proportional to $x \exp(ax)$ in the interval $(0, b)$.

Generate a uniform random number U;
$Y \leftarrow X(k_0 + U(k_1 - k_0))$.
`Else` *(The generator region is unbounded)*
We use the following method to sample from the marginal distribution:
Generate a uniform random number U;
$U \leftarrow U(|b| + |k_1 - k_0|/a)$.
`If` $(U < |b|)$,
$X \leftarrow -\log(U/|b|)/a$ *(X is an exponential random variate)*
`Else`
Generate two independent uniform random numbers U_1, U_2;
$X \leftarrow -\log(U_1 * U_2)/a$. *($X$ is a gamma(2) random variate:)*
Generate a uniform random number U;
$Y \leftarrow Xk_0 + U(b + X(k_1 - k_0))$.
$z \leftarrow s + Xa$. *(This is the value of the tangential plane.)*
$\tilde{X} \leftarrow x_t + r_1 X + r_2 Y$;
$\tilde{Y} \leftarrow y_t - r_2 X + r_1 Y$.
The back transformed pair $(\tilde{X}, \tilde{Y})$ is a sample from the hat function.
Generate a uniform random number V.
`If` $\log(V) \le \log(f(\tilde{X}, \tilde{Y})) - z$,
accept←true,
`Else`
accept←false.
`If` the number N of design points N has not been reached,
`and if` the aimed acceptance probability has not been reached,
`call` **add-new-points** with $(\tilde{X}, \tilde{Y})$ added as new point of contact.
`Until` accept.
`Return` the pair $(\tilde{X}, \tilde{Y})$ as a sample of the desired bivariate distribution.

To make the set-up for large N faster, it is better to consider the following variant of adaptive rejection sampling by changing the condition for calling Algorithm Add-new-points above:
If there are already ten design-points used for the hat, store pairs (X, Y) and start Add-new-points only to add (e.g.) 5 new design-points together. This change means that the set-up becomes considerably faster as Algorithm Add-new-points executes slowly for a larger number of design-points. The disadvantage is that the average acceptance probability is slightly lowered.

4. COMPUTATIONAL EXPERIENCE

First we want to address the complexity of Algorithm ALC2D in terms of the number of design points N. To do this we need the following considerations: First we count the total number of vertices of the partition of the whole plane. For three design points we clearly have four vertices: One true vertex and 3 vectors towards infinity. Adding one design point means adding two vertices, in special cases less than two vertices are added. Thus the total number of vertices of the partition of the whole plane can never exceed $2N-2$. (In the case of a bounded domain, we have to add the vertices of the domain of the distribution n_d but as this number is fixed and in most cases small we do not consider it further.) Using the same arguments as above we can see that the total number of polygon vertices is always smaller than $6N-6$ and the average number of vertices of one polygon can never exceed $6-6/N$. The total number of triangles after the triangulation is therefore smaller than $4N$ and the total number of generator regions can never exceed $8N$. Our experiments with the algorithm showed that the actual number of generator regions for a large number of design points often comes quite close to $8N$.

The execution time of Algorithm Sample2d does not grow with N as long as we can guarantee that the generation of the index of the generator region is not influenced by N. Using a guide-table method this is the case if the size of the guide-table grows linearly with the total number of generator regions or – which is the same – linearly with N.

Addressing the complexity of the set-up for N design-points we can see that all steps of algorithm Add-design-points have complexity $O(N)$. Only the body of the loop "for all other design points" calls algorithm Polyupdate N^2 times. Algorithm Polyupate's complexity grows linearly with the number of vertices of the polygon, and the average number of vertices is smaller than 6. Thus that part and the whole Algorithm Add-design-points has complexity $O(N^2)$.

We tested our C-implementation of Algorithm ALC2D for the bivariate normal distribution; for the minimum of a bivariate normal density and an exponential density of the form $exp(a+bx+cy)$ which we will call "cut-normal distribution"; for the bivariate beta distribution (cf. [Mauldon 1959]) with density proportional to

$$f(x,y)=x^{a_1-1}y^{a_2-1}(1-x-y)^{a_3-1} \quad \text{for} \quad x,y\geq 0, \quad x+y\leq 1 \qquad a_1,a_2,a_3\geq 1.$$

and for the following log-concave non-standard distributions:

$$f(x,y)=xe^{-x^2-xy-y^2} \quad \text{for} \quad x\geq 0 \quad \text{called NS1-distribution}$$

$$f(x,y)=min(e^{n^2-n\sqrt{x^2+y^2}},1) \qquad n>0 \quad \text{called NS2-distribution}$$

$$f(x,y)=e^{ax^4+bx+cxy+dy^2+ey^4} \qquad a,b,d,e>0 \qquad 4bd\geq c^2 \quad \text{called NS3-distribution}$$

For these six densities – also restricted to (open and closed) polygons – our extensive experiments showed that the algorithm works for a really wide range of parameters. For example using a sensible auxiliary domain and a starting value close to the mode, random variates from the normal distribution with $s_x=10^{12}$, $s_y=10^{-2}$ and $r=0.9999$ can be generated without problems. The NS2 density has a constant circle shaped plateau and goes fast to zero outside the circle. Therefore the auxiliary domain must contain the full plateau but should not be much larger. Used in that way the algorithm works with n up to 10^8. For the beta-distribution without auxiliary domain and the mode as starting value there occurred no problems for example for $a_1=20$, $a_2=6$, $a_3=10^{14}$.

Of course numerical problems can occur, if certain parameters are chosen very big, mainly because evaluating the density or the partial derivatives or the volume below the hat becomes numerically unstable.

To start the algorithm we used rectangles of different sizes as auxiliary domain and one starting value not too far away from the mode. (For the bivariate beta-distribution an auxiliary domain is not necessary.) Of course the acceptance probability P_N of our algorithm is a random variable. P_{100} was for all our experiments larger than 95.8 % . Among the tested distributions NS2 has the highest expected value of P_{100} (0.9987), the cut-normal distribution ranks second. The expectation of the normal, beta, NS1 and NS3 distribution are all close to 0.97. For all tested distributions the standard deviation of P_{100} is very small, in the order of 10^{-3}. Summarising these results we can say that P_{100} only slightly depends on the density and is closer to one if the logarithm of the density can be well approximated by a linear function. We tested these densities with $N = 100$ for several choices of auxiliary domain and starting value. It turned out that the influence of the auxiliary domain and the starting value is negligible. 100 design-points require a time-consuming set-up and should only be used if the needed sample-size is very large. Otherwise between twenty and fifty design points will be enough. For example P_{20} had expectations between 0.72 and 0.97 and standard deviations between 0.01 and 0.1 and these values are influenced not only by the distribution but also by the choice of the auxiliary domain and the starting values. Based on these results we suggest to use comparatively small auxiliary domains (eg. the rectangle $(-\sigma_X, \sigma_X) \times (-\sigma_Y, \sigma_Y)$ for the standard normal distribution and starting values close to the mode. We found it astonishing that the mode as starting value is not always leading to the best results; sometimes starting values close to the mode are better, but starting values far away from the mode are always worse.

In our algorithm the set-up and the generation of points are performed at the same time. The expected number of random pairs that must be generated to get one new point of contact is the number of trials till one pair is rejected and thus equal to $1/(1 - P_N)$. Therefore a large number of generated pairs till N contact points are reached is no disadvantage. It is only an indication that the acceptance probability is already close to one.

Now we compare our new algorithm with the recent universal algorithms for n-dimensional log-concave distributions. The algorithm of [Leydold and Hörmann 1998] only exists as a prototype and is more than ten times slower than our new algorithm, as it is not using the special properties of the R^2. The algorithm suggested in [Leydold 1998] can be applied to two-dimensional log-concave distributions as well but it is about two times slower than our method and also has the theoretical drawback that the acceptance probability does not converge to one but to a lower value, which depends on the distribution (eg. 0.73 for the bivariate standard normal distribution). It is difficult to compare Algorithm ALC2D with other existing algorithms as an automatic algorithm for bivariate log-concave distributions has not been developed before. The speed to generate a pair of the bivariate normal distribution (not considering the set-up) is depending on the computer and the uniform generator used only between 10 % and 70 % slower than the generation of an independent normal pair with the well known Box-Muller (also called sine-cosine) method. Applied to the bivariate beta-distribution Algorithm ALC2D is, due to the density which is more expensive to evaluate, about 40 % slower than for the normal distribution, for the other tested distributions the execution times lie in between. The set-up is of course quite slow. It is difficult to separate its time from the generation time as random pairs are generated during the set-up as well, but roughly speaking for $N = 50$ the set-up takes about the time of generating 10000 pairs.

Our computational experience shows that Algorithm ALC2D can generate variates of bivariate log-concave distribution over (possibly open) convex polygons. This is quite a lot if we compare it with the fact that even for the task of generating pairs of the bivariate normal distribution restricted to a convex polygon no general method is discussed in the literature.

ACKNOWLEDGMENTS

I would like to thank Professor G. Derflinger from the Department of Statistics of the University of Economics and Business Administration Vienna, for supporting me in so many ways during the many years I worked on this paper.
I also would like to thank Professor A.R. Kaylan who made it possible that I could finish this work in Istanbul.

REFERENCES

Ahrens, J. H. 1995. An one-table method for sampling from continuous and discrete distributions. *Computing 54*, 2, 127–146.

Dagpunar, J. 1988. *Principles of Random Variate Generation.* Clarendon Oxford Science Publications, Oxford, U.K.

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

Devroye, L. 1997. Random variate generation for multivariate densities. *ACM Transactions on Modeling and Computer Simulation 7*, 4, 447–477.

Evans, M. and Swartz, T. 1998. Random variable generation using concavity properties of transformed densities. *Journal of Computational and Graphical Statistics 7*, 4, 514–528.

Gilks, W. R. and Wild, P. 1992. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics 41*, 337–348.

Hörmann, W. 1995. A rejection technique for sampling from T-concave distributions. *ACM Transactions on Mathematical Software 21*, 2, 182–193.

Hörmann, W. and Deflinger, G. 1994. Universal generators for correlation induction. In R. Dutter and W. Grossmann Eds., *Compstat, Proceedings in Computational Statistics* (Heidelberg, 1994), pp. 52–57. Physica-Verlag.

Johnson, M. E. 1987. *Multivariate Statistical Simulation.* Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. John Wiley & Sons, New York.

Leydold, J. 1998. A rejection technique for sampling from log-concave multivariate distributions. *ACM Transactions on Modeling and Computer Simulation 8*, 3, 254–280.

Leydold, J. and Hörmann, W. 1998. A sweep-plane algorithm for generating random tuples in simple polytopes. *Mathematics of Computation 67*, 224, 1617–1635.

Mauldon, J. G. 1959. A generalization of the beta-distribution. *Annals of Mathematical Statistics 30*, 509–520.

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

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