***S**tatistica & **A**pplicazioni*
Vol. II, n. 2, 2004
pp. 25-44

Random variate generation from Multivariate Exponential Power distribution

Nadia Solaro[§]

Summary:
In this paper two procedures for random variate generation for the family of Multivariate Exponential Power (MEP) distributions are developed and then compared. By conveniently expressing MEP distributions in terms of mixture of random variables, a simple and fast procedure based on transformation approach is proposed. Another procedure, based on a general acceptance-rejection (AR) technique for multivariate orthounimodal densities, is considered. As expected, the transformation-based procedure is considerably less time-consuming than the other one. Furthermore, this study allows us to evaluate to which extent generalizations of AR methods could be computationally less efficient than a transformation one for the type of multivariate distributions considered.

Keywords: *acceptance-rejection method, multivariate elliptical and symmetric distributions, multivariate orthounimodal distributions, transformation method.*

1. Introduction

As known, there is not yet a large literature on Monte Carlo methods for multivariate random variate generation. Johnson's monograph (1987) remains perhaps the most important reference, though new efforts in this research field have recently been made. Actually, approaches to the problem of multivariate random variate generation are mainly based on multivariate extensions of generation methods for univariate random variables, such as ratio-of-uniforms method, acceptance-rejection (AR) methods or transformation method. A more typical multidimensional approach is the conditional distribution method (Johnson, 1987).

[§] Dipartimento di Statistica – Università degli Studi di Milano Bicocca – via Bicocca degli Arcimboldi, 8, 20126 MILANO (e-mail: nadia.solaro@unimib.it).

On the other hand, all of the mentioned methods usually present difficulties in the multidimensional framework. For instance, conditional distribution approach may be impractical for many multivariate distributions, as it requires sampling from the marginal as well as the conditional distributions, which do not always exist in closed form. As regards ratio-of-uniforms method, it seems still unclear for most multivariate distributions how to determine the hyperrectangle that defines the acceptance region. The main difficulty in the application of AR methods is the construction of dominating functions (also called "hat functions"), through which the acceptance/rejection region is outlined. The most convenient approach is probably the transformation method, which generally results in ease to develop and computationally efficient algorithms. Nevertheless, for many multivariate distributions it is often impossible to define manageable transformations of random vectors from which samples can be easily obtained. Linear transformations of either multivariate normal distribution or random vectors uniformly distributed on the surface of hyperspheres in $\Re^n$ are notable exceptions. For more details on these methods see, for instance, Rubinstein (1981), Devroye (1986) and Johnson (1987).

The main aim of this study is to develop procedures for generating from a particular family of multivariate distributions, namely the family of Multivariate Exponential Power distributions, (MEPs). A first procedure is based on the transformation approach; it results in a very simple and fast algorithm as a consequence of a link outlined between MEPs and random vectors uniformly distributed on the surface of hyperspheres in $\Re^n$. Empirical evaluations of the performance of the proposed algorithm are also considered.

The transformation-based method is then compared with another procedure developed on the basis of a general AR method proposed by Devroye for multivariate orthounimodal distributions (Devroye, 1997). As Johnson stated (Johnson, p.46, 1987), "The AR methods… have not had much impact in multivariate generation", mainly because of the difficulty in finding a dominating function. Recently, research in this direction has shown that if some additional information on the multivariate densities is available, then AR techniques could be more successfully applied. On this matter, Devroye's work (1997) is the key reference. Indeed, our transformation-based procedure is expected to be computationally more efficient than the AR method. Anyway, it seems interesting to evaluate empirically to which extent an AR algorithm could be computationally less convenient than a transformation algorithm in the context of MEP distributions.

The need to generate from the family of MEP distributions arises essentially from the fact that this family is one of the possible generalizations of multivariate normal distribution in terms of ellipsoidal departures. In this respect, MEP distributions could be particularly appealing in studies about robustness of estimation procedures when normality assumption is violated.

Recently, MEP distributions have received particular attention as the distribution laws of random effects in a multilevel model with the goal of examining robustness of maximum likelihood estimation procedures for model parameters, (Solaro, Ferrari, 2003a, 2003b).

This paper is organized as follows. In Section 2, principal characteristics of MEP distributions are briefly recalled, such as probability density function and role of parameters. A subsection contains some basic results on multivariate elliptical and symmetric family of distributions, to which MEPs belong. Section 3 contains a synthetic description of Devroye's AR methods for generating from multivariate orthounimodal distributions, whose class includes MEP distributions as a special case. In order to improve the construction of hat function, and consequently to reduce the rejection region, we exploited symmetry of MEPs as the extra information on density function. Successive sections are the core of this work. In Section 4, two new procedures for generating from MEPs are introduced. Section 5 reports experimental results. The two procedures have been implemented in R language, vs. 1.8.1, the well-known integrated environment for both programming and statistical data analysis, (R Development Core Team, 2003). Section 6 is dedicated to conclusions. Finally, computation details and R programming codes are gathered in Appendices A and B.

2. Multivariate exponential power distributions

The n-dimensional random vector (r.v.) $\boldsymbol{X}$ is $\text{MEP}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \kappa)$ if its density function (d.f.) may be expressed as follows:

$$f(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma},\kappa) = \frac{n\Gamma\left(\frac{n}{2}\right)}{\pi^{\frac{n}{2}}\Gamma\left(1+\frac{n}{\kappa}\right)2^{1+\frac{n}{\kappa}}\left|\boldsymbol{\Sigma}\right|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}\left[\left(\mathbf{x}-\boldsymbol{\mu}\right)^T\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}\right)\right]^{\frac{\kappa}{2}}\right\}, \quad (1)$$

where $\boldsymbol{\mu} \in \Re^n$, $\kappa > 0$ and $\boldsymbol{\Sigma}$ is an $n \times n$ positive definite matrix. Mean vector is: $\mathrm{E}(\boldsymbol{X}) = \boldsymbol{\mu}$ and variance-covariance matrix is: $\mathrm{V}(\boldsymbol{X}) = c(\kappa, n)\,\boldsymbol{\Sigma}$, being $c(\kappa, n) = 2^{2/\kappa}\Gamma((n+2)/\kappa)/(n\Gamma(n/\kappa))$. By substituting in (1) null vector for $\boldsymbol{\mu}$ and identity matrix for $\boldsymbol{\Sigma}$, the so-called spherical MEP distributions are obtained.

As κ varies a specific member of the family of MEP distributions is identified. As special cases, for $\kappa = 1$ the multivariate Laplace distribution is obtained, while for $\kappa = 2$ the multivariate normal distribution is derived. Since it actually represents kurtosis departure from multivariate normal distribution, κ is also called the "non-normality" parameter: in particular, for $\kappa < 2$ and $\kappa > 2$ respectively leptokurtic and platikurtic multivariate

distributions are obtained. Figure 1 illustrates density plots for six bivariate and spherical exponential power distributions and for some values of κ.

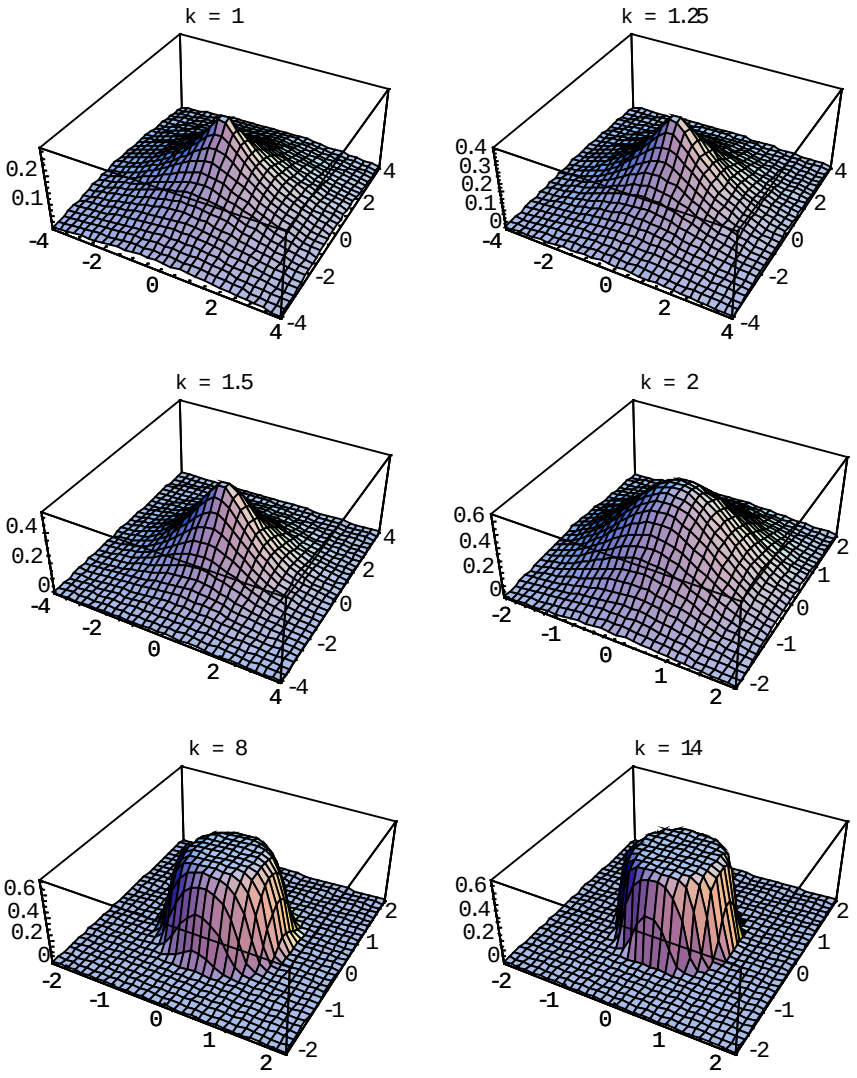

Figure 1. *Density plots for six bivariate and spherical exponential power distributions with the non-normality parameter* $\kappa = 1, 1.25, 1.5, 2, 8, 14$.

2.1 *MEP distributions in the context of elliptical distributions*

MEP distributions belong to multivariate Kotz family, which is a particular class of symmetric and elliptical distributions (Fang *et al.*, 1990). Among several equivalent definitions of elliptical distribution, we consider the following one: the *n*-dimensional r.v. $\boldsymbol{S}$ is said to have an elliptically contoured, or simply elliptical, distribution with parameters $\boldsymbol{\mu}$ ($n \times 1$) and $\boldsymbol{\Sigma}$ ($n \times n$) if its d.f. is of the following form:

$$f_{\boldsymbol{S}}(\mathbf{s};\boldsymbol{\mu},\boldsymbol{\Sigma}) = |\boldsymbol{\Sigma}|^{-\frac{1}{2}} g\left[(\mathbf{s}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{s}-\boldsymbol{\mu})\right], \tag{2}$$

for some real-valued function g and with $\boldsymbol{\Sigma}$ positive definite. We shall denote $\boldsymbol{S} \sim \mathrm{EC}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, g)$, where EC is the abbreviation for "Elliptically Contoured". The parameter $\boldsymbol{\mu}$ represents the mean vector, $\boldsymbol{\Sigma}$ is called "characteristic matrix" and g is the density generator, i.e. the function that identifies a particular family in the class of EC distributions. By putting in (2) $\boldsymbol{\mu}$ equal to the null vector and $\boldsymbol{\Sigma}$ to the identity matrix, a spherically distributed r.v. $\boldsymbol{V}$ is obtained. In brief: $\boldsymbol{V} \sim \mathrm{EC}_n(\mathbf{0}, \mathbf{I}, g)$.

Multivariate Kotz type distributions possess density generator of the form:

$$g(q) = c_n q^{H-1} \exp(-hq^m), \tag{3}$$

where $h > 0$, $m > 0$, $2H + n > 2$, c_n is a normalizing constant and q denotes the quadratic form $(\mathbf{s}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{s}-\boldsymbol{\mu})$. MEP distributions are then elliptical since their d.f. (1) is of the form (2) and their density generator may be written in the form (3) with $H = 1$, $h = 1/2$, $m = \kappa/2$ and normalizing constant $c_n = C_{n,\kappa} = n\Gamma\left(\frac{n}{2}\right)/\pi^{\frac{n}{2}}\Gamma\left(1+\frac{n}{\kappa}\right)2^{1+\frac{n}{\kappa}}$.

For our generating purposes, the next results are relevant. They are extensively treated in Fang, Kotz, Ng's monograph (1990).

Firstly, any linear combination of elliptically distributed random vectors is still elliptical with the same density generator g. Formally, if $\boldsymbol{S} \sim \mathrm{EC}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, g)$, $\mathbf{a}$ is a $p \times 1$ vector and $\mathbf{B}$ is an $n \times p$ matrix, then:

$$\mathbf{a} + \mathbf{B}^T \boldsymbol{S} \sim \mathrm{EC}_p(\mathbf{a} + \mathbf{B}^T\boldsymbol{\mu}, \mathbf{B}^T\boldsymbol{\Sigma}\mathbf{B}, g). \tag{4}$$

See Fang *et al.* (1990) for a proof.

Secondly, the r.v. $\boldsymbol{S}$ is $\mathrm{EC}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, g)$, with $\mathrm{rank}(\boldsymbol{\Sigma}) = n$, if and only if:

$$\boldsymbol{S} \overset{\mathrm{d}}{=} \boldsymbol{\mu} + R\,\mathbf{A}^T \boldsymbol{U}^{(n)}, \tag{5}$$

where the univariate random variable R is non negative and is stochastically independent of the r.v. $\boldsymbol{U}^{(n)}$, uniformly distributed on the unit hypersphere surface in R^n. Moreover, $\mathbf{A}$ is an $n \times n$ matrix such that $\mathbf{A}^T\mathbf{A} = \boldsymbol{\Sigma}$.

Expression (5), also known as Cambanis, Huang and Simons's stochastic representation, defines the relation through which the distribution law of an elliptical r.v. $\boldsymbol{S}$ links always with the uniform distribution of the r.v. $\boldsymbol{U}^{(n)}$, while it is a suitable random variable R that identifies a specific family in the class of EC distributions.

The d.f. of R is known. If $\boldsymbol{V} \sim \mathrm{EC}_n(\mathbf{0}, \mathbf{I}, g)$ then the d.f. $f_R(r)$ of R is:

$$f_R(r) = \frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} r^{n-1} g(r^2), \tag{6}$$

see Fang *et al.* (1990). Moreover, R^2 has the same distribution as the quadratic form $\boldsymbol{V}^T\boldsymbol{V} = \left\|\boldsymbol{V}\right\|^2$, where $\|\cdot\|$ denotes the Euclidean norm.

3. General rejection technique for multivariate orthounimodal distributions

Recently, Devroye (1997) has developed general procedures for simulating multivariate orthounimodal (MOr) distributions which possess a d.f.. MOr density is a d.f. $f(\mathbf{x})$ on $\Re^n$ with mode at $\mathbf{m} = (m_1, ..., m_n)$ for which for each x_i, $f(\mathbf{x})$ is a nonincreasing function of x_i as x_i tends to $+\infty$ ($x_i \geq m_i$) and as x_i tends to $-\infty$ ($x_i \leq m_i$), with all other x_j components fixed, $j \neq i$, $j, i = 1, ..., n$. In other terms, a d.f. $f(\mathbf{x})$ is orthounimodal if it is a monotone function of each single argument x_i ($i = 1, ..., n$) within each of the 2^n quadrants Q that are formed in $\Re^n$ at the mode $\mathbf{m}$. We may also say that the d.f. $f(\mathbf{x})$ is orthomonotone over each Q (Devroye, 1997). Without loss of generality, we shall assume that the mode $\mathbf{m}$ is at the origin ($\mathbf{m} = \mathbf{0}$).

In order to simulate MOr densities it suffices to work on positive quadrant $Q^+ \subset \Re^n$ only, that is: $Q^+ = \{\mathbf{x} \in \Re^n \mid x_i \geq 0 \ \forall \ i = 1, ..., n\} = [0, \infty)^n$. Then, generating on all Qs requires to suitably attribute the sign randomly to each component of the vector $\mathbf{x}^*$ previously generated on Q^+ (see Section 8 in Devroye, 1997).

Devroye's general procedures for MOr densities are based on Von Neumann's acceptance-rejection method. In general, in multidimensional framework AR technique consists of simulating a d.f. $f(\mathbf{x})$ by employing inequalities of the form $f(\mathbf{x}) \leq \mathrm{K}\eta(\mathbf{x})$, for all $\mathbf{x} \in \Re^n$, where K is a constant and $\eta(\mathbf{x})$ is a d.f. on $\Re^n$ from which it is ease to generate. Then, random

variate generation from $f(\mathbf{x})$ may be carried out through algorithms of the type described in Table 1, (Johnson, 1987).

In order to employ AR algorithms for generation from MOr densities Devroye suggests choosing the hat function $\mathrm{K}\eta(\mathbf{x})$ on Q^+ among some basic families of distributions. Such a choice can be based on some further information on the d.f. $f(\mathbf{x})$. In particular, if in addition $f(\mathbf{x})$ is symmetric in all x_is, then the basic family to be referred to is that of Multivariate Max (MMax) distributions. These are distributions with d.f. of the form $\varphi(\max_{1\le i\le n} x_i)$ for some functions φ on Q^+.

Table 1. *General AR algorithm for Multivariate Distributions*

Repeat	
Step 1	generate a value u^* from $\mathcal{U}$ uniform on [0, 1];
Step 2	generate $\mathbf{w}^*$ from r.v. $\boldsymbol{W} \sim \eta(\mathbf{w})$ on $\Re^n$;
Step 3	if $u^*\mathrm{K}\eta(\mathbf{w}^*) \le f(\mathbf{w}^*)$, return $\mathbf{w}^* = \mathbf{x}^*$ as a value of $\boldsymbol{X} \sim f(\mathbf{x})$; otherwise, go to Step 1.

Here we devote particular attention to the case of MOr and symmetric densities, since MEP distributions share both these properties. Therefore, Devroye's method for simulation of a MMax density plays a central role. It is based on generating from random variable $X_{\mathrm{M}} = \max\{X_1, \ldots, X_n\}$, whose d.f. is given by:

$$f_{X_M}(x_M) = nx_M^{n-1}\varphi(x_M), \tag{7}$$

($x_M > 0$); the d.f. (7) is also called density generator for $\varphi(\max_{1\le i\le n} x_i)$ (for more details see Devroye, 1997). Devroye's algorithm for generating from MMax distributions is described in Table 2.

Table 2. *Devroye's algorithm for Multivariate Max Distributions*

Repeat	
Step 1	generate a value x_M^* from X_M with d.f. (7);
Step 2	generate n values u_i^* from n i.i.d. $\mathcal{U}_i$ uniform on [0, 1];
Step 3	generate an integer w^* from $\mathcal{W}$ uniform on {1, …, n};
Step 4	set $u_{w^*}^* = 1$;
Step 5	$\mathbf{x}^* = (x_M^* u_1^*, \ldots, x_M^* u_n^*)$ is a value from $\boldsymbol{X}$ with d.f. proportional to $\varphi(\max_{1\le i\le n} x_i)$.

Step 5 in Table 2 is justified by the following result. If $\boldsymbol{X}$ is a r.v. with d.f. of the type $\varphi(\max_{1\le i\le n} x_i)$, then the r.v. $(X_M \mathcal{U}_1, \ldots, X_M \mathcal{U}_n)$ has d.f. proportional to $\varphi(\max_{1\le i\le n} x_i)$, where $X_M = \max\{X_1, \ldots, X_n\}$ with d.f. (7), $\mathcal{U}_i$s are n i.i.d. uniformly on [0, 1], except $\mathcal{U}_w$ which equals 1 and $\mathcal{W}$ is uniformly distributed on {1, …, n}. Moreover, given $X_M = X_i$, the remaining X_js are i.i.d. uniformly on [0, X_M].

Hence, for MOr and symmetric densities $f(\mathbf{x})$ one can apply AR technique by employing inequalities of this type: $f(\mathbf{x}) \le h(\max_{1\le i\le n} x_i)$, where the hat function $h(\max_{1\le i\le n} x_i)$ is proportional to a MMax density $\varphi(\max_{1\le i\le n} x_i)$ and $h(t) = f(t, 0, 0, \ldots, 0)$, (Devroye, 1997). Then, we shall write:

$$f_{X_M}(x_M) \propto n x_M^{n-1} h(x_M) \tag{8}$$

and a general AR algorithm (Table1) for MOr and symmetric distributions is reported in Table 3.

Table 3. *Devroye's AR algorithm for MOr and symmetric distributions*

Repeat	
Step 1	generate a value u^* from $\mathcal{U}$ uniform on [0, 1];
Step 2	generate $\mathbf{x}^*$ from r.v. $\boldsymbol{X}$ with MMax d.f. $\varphi(\max_{1\le i\le n} x_i) = c_n h(\max_{1\le i\le n} x_i)$, where c_n is the normalizing constant (algorithm in Table 2);
Step 3	if $u^* h(\max_{1\le i\le n} x_i) \le f(\mathbf{x}^*)$, return $\mathbf{x}^*$ as a value from $\boldsymbol{X} \sim f(\mathbf{x})$ on quadrant Q^+ and then suitably attribute a random sign to each component of $\mathbf{x}^*$; otherwise, go to Step 1.

Note that the constant c_n in Step 2 of Table 3 represents the inverse of the expected number $\mathcal{K}_n$ of trials required for generating each value $\mathbf{x}^*$, that is:

$$\mathcal{K}_n = \frac{1}{c_n} = \int_0^{+\infty} n x_M^{n-1} h(x_M)\, dx_M \ , \tag{9}$$

(Devroye, 1997).

4. Two methods for generation from MEP distributions

We first develop a fast procedure for generating from a r.v. $\boldsymbol{X} \sim \text{MEP}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \kappa)$. Following Johnson's hints (1987), methods for generating random variates from elliptical distributions can be based on Cambanis, Huang and Simons's stochastic representation (5), the key of the transformation approach. Our procedure (Method A) is then developed thereby.

Secondly, with comparative aims we develop a further procedure (Method B), which is based on the general Devroye's rejection algorithm for MOr and symmetric densities (Table 3).

The previous result (4) allows us to focus on generating from spherical r.v. $\boldsymbol{Y} \sim \text{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$ only. Then, for generating from r.v. $\boldsymbol{X} \sim \text{MEP}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \kappa)$ a linear transformation of $\boldsymbol{Y}$ can be considered, namely: $\boldsymbol{X} = \boldsymbol{\mu} + \mathbf{A}^T \boldsymbol{Y}$, where $\mathbf{A}$ is an $n \times n$ matrix such that: $\mathbf{A}^T \mathbf{A} = \boldsymbol{\Sigma}$. A decomposition of matrix $\boldsymbol{\Sigma}$ can be achieved, for instance, through Cholesky factorization, which is known to be computationally fast (Thisted, 1988).

4.1 *Method A*

The above statements (5)-(6) can be specified to the case of the r.v. $\boldsymbol{Y} \sim \text{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$. More precisely, we note that $\boldsymbol{Y} \overset{\text{d}}{=} R\boldsymbol{U}^{(n)}$ with R distributed according to the κ-th root of a Gamma with shape parameter n/κ and scale parameter $1/2$, that is:

$$R \overset{\text{d}}{=} \sqrt[\kappa]{\text{Gam}\left(\frac{n}{\kappa}, \frac{1}{2}\right)}, \tag{10}$$

(Appendix A, Section A.1).

In order to generate from MEP, we can then use the stochastic representation (5) along with the fact that methods for generating from r.v. $\boldsymbol{U}^{(n)}$, uniformly distributed on the unit hypersphere surface in $\Re^n$, are available. Here we focus on Box-Muller's method, according to which if $\boldsymbol{Z}$ is a multivariate normal with standardized components, then the $\boldsymbol{U}^{(n)}$ can be obtained by:

$$\boldsymbol{U}^{(n)} \overset{\text{d}}{=} \frac{\boldsymbol{Z}}{\|\boldsymbol{Z}\|}, \tag{11}$$

(Johnson, 1987).

Hence, for generating from $\boldsymbol{Y} \sim \text{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$ with n and κ fixed, the algorithm described in Table 4 can be implemented.

Table 4. *Algorithm A for MEP distributions (Transformation method)*

Repeat	
Step 1	generate a value r^* from $R = \sqrt[\kappa]{\text{Gam}\left(\frac{n}{\kappa}, \frac{1}{2}\right)}$;
Step 2	generate n values z_i^* from multivariate normal $\boldsymbol{Z} \sim N_n(\mathbf{0}, \mathbf{I})$;
Step 3	generate $\mathbf{u}^*$ from $\boldsymbol{U}^{(n)}$ by using (11);
Step 4	$\mathbf{y}^* = r^* \mathbf{u}^*$ is a value from $\boldsymbol{Y} \sim \text{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$.

4.2 *Method B*

We have already noted that MEP distributions are both MOr and symmetric. We start working on positive quadrant $Q^+ = [0, \infty)^n$. Implementation of Devroye's method requires first to determine the d.f. of $Y_M = \max\{Y_1, \ldots, Y_n\}$, with $Y_i \geq 0$ for all i. By substituting in (8): $h(y_M) = C_{n,\kappa} \exp\left\{-\frac{1}{2} y_M^{\kappa}\right\}$,

where $y_M > 0$ and $C_{n,\kappa} = n\Gamma\left(\frac{n}{2}\right)/\pi^{\frac{n}{2}}\Gamma\left(1+\frac{n}{\kappa}\right)2^{1+\frac{n}{\kappa}}$, we note that Y_M is distributed as the κ-th root of a Gamma with shape parameter n/κ and scale parameter $1/2$, that is:

$$Y_M \overset{d}{=} \sqrt[\kappa]{\mathrm{Gam}\left(\frac{n}{\kappa}, \frac{1}{2}\right)}, \tag{12}$$

(Appendix A, Section A.2). Therefore, we can refer to the r.v. $(Y_M\mathcal{U}_1,\ldots,Y_M\mathcal{U}_n)$ with d.f. proportional to $\varphi(\max_{1\le i\le n} y_i)$, whose components could be analogously described as made in Section 3.

Secondly, for the acceptance-rejection test, we observe that inequality in Step 3 of Table 3 in the present case reduces to:

$$u^* \exp\left\{-\tfrac{1}{2} y_M^{*\kappa}\right\} \le \exp\left\{-\tfrac{1}{2}\left\|\mathbf{y}^*\right\|^{\frac{\kappa}{2}}\right\}, \tag{13}$$

where $\mathbf{y}^* = (y_1^*,\ldots,y_n^*)$ denotes a value from the r.v. $(Y_M\mathcal{U}_1,\ldots,Y_M\mathcal{U}_n)$ and $y_M^* = \max(y_1^*,\ldots,y_n^*)$.

Moreover, given that (13) holds and then $\mathbf{y}^*$ is accepted as a value from $\boldsymbol{Y} \sim \mathrm{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$ on the positive quadrant Q^+, a sign to each component y_i^* of $\mathbf{y}^*$ can be assigned by simply multiplying by −1 or +1, randomly generated. In fact, for MOr and symmetric d.f. $f(\mathbf{x})$, if $f(\mathbf{x}) \le \mathrm{K}\eta(\mathbf{x})$ holds for all $\mathbf{x} \in Q^+$, then by symmetry $f(\mathbf{x}) \le \mathrm{K}\eta(|\mathbf{x}|)$ holds for all $\mathbf{x} \in \Re^n$, where by definition $|\mathbf{x}| = (|x_1|, |x_2|, \ldots, |x_n|)$, (Devroye, 1997, Section 8).

In conclusion, for generating from MEP distributions, with n and κ fixed, the algorithm described in Table 5 can be referred to.

Table 5. *Algorithm B for MEP distributions (AR method)*

Repeat	
Step 1	generate a value u^* from $\mathcal{U}$ uniform on [0, 1];
Step 2	generate $\mathbf{y}^*$ from $\boldsymbol{Y}$ with MMax d.f. proportional to $\varphi(\max_{1\le i\le n} y_i)$ on quadrant Q^+ (algorithm in Table 2 with expression (12));
Step 3	if inequality (13) holds, return $\mathbf{y}^*$ as a value from $\boldsymbol{Y} \sim \mathrm{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$ on quadrant Q^+ and assign randomly a sign to each component y_i^* of $\mathbf{y}^*$ as previously described; otherwise, go to Step 1.

As to the expected number $\mathcal{K}_n$ of iterations, it is ease to see that here formula (9) reduces to:

$$\mathcal{K}_n = \frac{\Gamma\left(1+\frac{n}{2}\right)}{\Gamma^n\left(\frac{1}{2}\right)} = \frac{\Gamma\left(1+\frac{n}{2}\right)}{\pi^{n/2}}. \tag{14}$$

Function $\mathcal{K}_n$ in (14) is decreasing with respect to n for $n \leq 5$ and increasing for $n > 5$ (see Figure 2). Consequently, the algorithm B results relatively efficient when the number of variables considered is 5.

5. Evaluation of performances of the two algorithms

Both the two algorithms A and B (Section 4) have been implemented in R language, vs. 1.8.1, and simulations have been carried out on a personal computer with processor speed 733 MHz. The univariate generation needed in both the algorithms has been achieved by relying on R functions `runif`, `rnorm` and `rgamma`. Programs are reported in Appendix B.

In order to evaluate the performance of the algorithm A (Table 4) we have considered generation from $\boldsymbol{Y} \sim \text{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$ with $n = 2; 3; 5; 8; 10; 15; 20; 25; 30$ variables and with $\kappa = 1; 1.25; 1.5; 1.75; 2; 5; 8; 11; 14$. For each pair (n, κ), 1,000 replications of 5,000 values per random variate have been generated and the execution times (in seconds) have been controlled through R function `system.time`. Average times have then been computed for each pair (n, κ); they are displayed in Table 6.

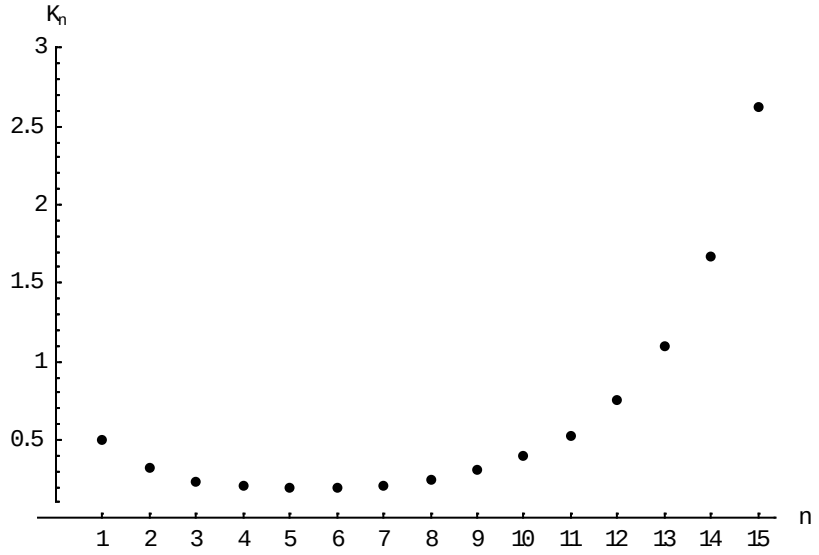

Figure 2. *Plot of expected numbers $\mathcal{K}_n$ of trials as a function of number n of variables. Numerical minimization leads to a minimum value for $\mathcal{K}_n$ (≈ 0.1895) approximately at the point 5.2570.*

Table 6 also reports some summary statistics for magnitude and variability. With regard to magnitude, general average times ($\bar{t}_n$) are presented for all *n* as well as average differences per additional variable (RDMean), i.e. the quantities $(\bar{t}_{n+a} - \bar{t}_n)/a$, where $\bar{t}_{n+a}$ and $\bar{t}_n$ denote general average times for generating from MEP with, respectively, $n + a$ and n variables, a integer. This latest index gives the mean effect size per additional variable from *n* to $n + a$ on general average times.

As for variability, Monte Carlo variance within-κ-values (W Var) and variance between-κ-values (B Var), percentage of within sum of squares on total sum of squares (% D_W/D_T) are shown for all *n*.

By inspection of RDMean index in Table 6 we see that generating 5,000 values per random variate involves rather constant general average execution times per additional variable, which are included between nearly 5 and 7 hundredth of second. Then, algorithm A reveals to be particularly fast.

Moreover, we note that for each pair (*n*, κ) the execution times are quite close to their general average time $\bar{t}_n$. In particular, for each *n* negligible values of variance within-κ-values (W Var) indicate that the execution times are only slightly spread around their average time; in other terms, they turn out to be rather stable over the 1,000 replications carried out for each κ. For *n* fixed, differences between average times computed for each κ value are

also negligible, as shown by very small variances between-κ-values (B Var).

Table 6. *Average execution times (in seconds) over 1,000 replications of 5,000 values per random variate generated by Algorithm A and for different pairs (n, κ).*

n / κ	*2*	*3*	*5*	*8*	*10*	*15*	*20*	*25*	*30*
1	0.410	0.470	0.597	0.808	0.949	1.195	1.614	1.867	2.408
1.25	0.437	0.501	0.630	0.846	0.973	1.219	1.650	1.894	2.199
1.5	0.451	0.503	0.629	0.846	0.974	1.320	1.647	1.894	2.187
1.75	0.450	0.503	0.627	0.842	0.973	1.319	1.637	1.899	2.187
2	0.453	0.501	0.746	0.844	0.973	1.319	1.638	1.898	2.186
5	0.451	0.498	0.639	0.858	0.975	1.308	1.659	1.901	2.196
8	0.453	0.499	0.637	0.852	0.976	1.308	1.647	1.904	2.215
11	0.452	0.501	0.635	0.837	0.977	1.315	1.638	1.920	2.187
14	0.452	0.499	0.638	0.859	0.978	1.315	1.641	2.004	2.187
$\bar{t}_n$	0.445	0.497	0.642	0.844	0.972	1.291	1.641	1.909	2.217
RDMean	-----	0.052	0.073	0.067	0.064	0.064	0.070	0.054	0.062
W Var	0.005	0.006	0.022	0.022	0.022	0.024	0.025	0.004	0.012
B Var	2e-04	1e-04	0.002	2e-04	8e-05	0.002	1e-04	0.001	0.005
%D_W/D_T	100	100	99.99	100	100	99.99	100	99.97	99.96

Percentages of within deviance on total deviance (% D_W/D_T) indicate the within variability to be essentially the only variability source for all *n* considered. For *n* fixed, the most dispersion is then due to slightly different execution times over various replications rather than to differences among average times for each κ.

As an illustration, Figure 3 shows plots in case of generation of 5,000 values per random variate from a spherical MEP with $n = 3$ variables and $\kappa = 8$. Diagonal panels report marginal distributions histograms; upper panels, bidimensional scatterplots of generated values; lower panels, smoothing bivariate density plots, (Bowman and Azzalini's R library `sm`).

Successively, with the aim to compare the algorithms A and B in terms of computational efficiency, we generated 1,000 replications of a single value per random variate from spherical MEPs with different combinations of *n* and κ, i.e. $n = 2; 3; 5; 7; 9; 11$ and $\kappa = 1; 1.5; 2; 5; 14$. Results appear in Table 7 as well as some summary statistics, such as general average times for both the algorithms (respectively, $\bar{t}_n^{\,A}$ and $\bar{t}_n^{\,B}$), and the relative efficiency of

algorithm B with respect to algorithm A, defined as: REff = $\bar{t}_n^{\mathrm{B}} \,/\, \bar{t}_n^{\mathrm{A}}$.

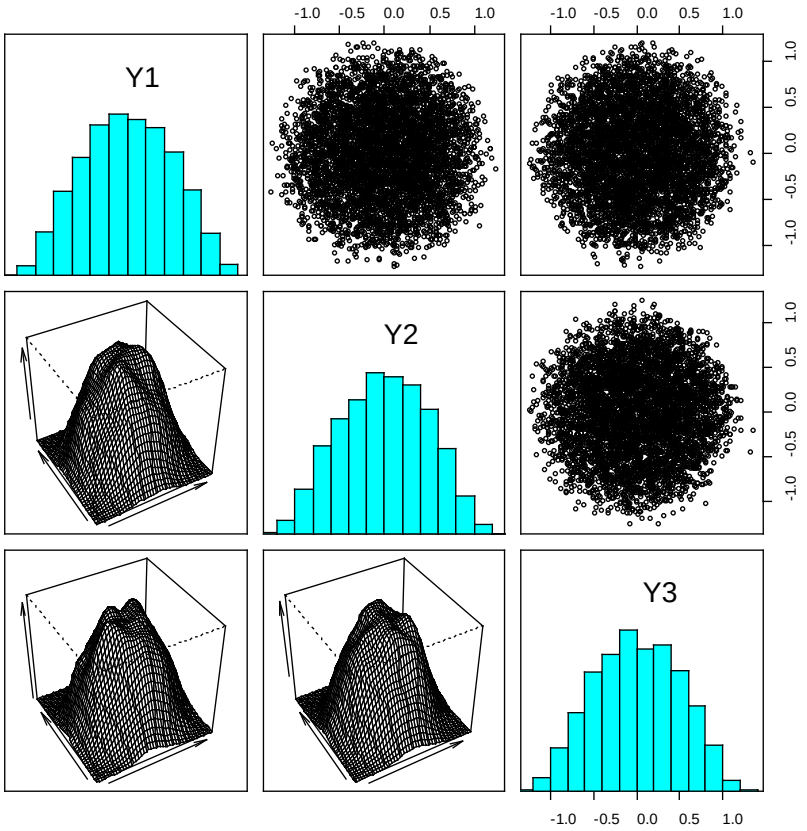

Figure 3. *Histograms (diagonal panels), scatterplots (upper panels) and smoothing bivariate density plots of estimated density (lower panels) in case of generation from a spherical MEP with* $n = 3$ *and* $\kappa = 8$ *(Algorithm A).*

Moreover, for algorithm B other two summary indices have been computed, referring to generation from spherical MEPs with $n + a$ variables with respect to those with n variables. The first index is the mean effect size per additional variable on general average times (RDMeanB), as already defined for Table 6. The second is the relative efficiency: REffB = $\bar{t}_{n+a}^{\mathrm{B}} \,/\, (a\,\bar{t}_n^{\mathrm{B}})$ of $n + a$ variables with respect to n, with a integer.

Table 7. *Average execution times (in seconds) over 1,000 replications of a single value per random variate generated by employing algorithms A and B and for different pairs (n, κ).*

κ	***n = 2***		***n = 3***		***n = 5***		***n = 7***		***n = 9***		***n = 11***	
1	A	0.0012	A	0.0012	A	0.0013	A	0.0013	A	0.0013	A	0.0013
	B	0.0022	B	0.0027	B	0.0068	B	0.0267	B	0.1484	B	1.0860
1.5	A	0.0012	A	0.0013	A	0.0019	A	0.0012	A	0.0013	A	0.0014
	B	0.0022	B	0.0028	B	0.0087	B	0.0298	B	0.1549	B	1.0631
2	A	0.0012	A	0.0013	A	0.0013	A	0.0013	A	0.0013	A	0.0014
	B	0.0023	B	0.0027	B	0.0069	B	0.0280	B	0.1496	B	1.1117
5	A	0.0012	A	0.0013	A	0.0013	A	0.0014	A	0.0014	A	0.0014
	B	0.0022	B	0.0032	B	0.0070	B	0.0290	B	0.1620	B	1.1537
14	A	0.0012	A	0.0013	A	0.0013	A	0.0013	A	0.0013	A	0.0014
	B	0.0022	B	0.0029	B	0.0068	B	0.0271	B	0.1569	B	1.1358
$\bar{t}_n^{\mathbf{A}}$	A	0.0012	A	0.0013	A	0.0014	A	0.0013	A	0.0013	A	0.0014
$\bar{t}_n^{\mathbf{B}}$	B	0.0022	B	0.0028	B	0.0072	B	0.0281	B	0.1543	B	1.1101
REff		1.8333		2.1538		5.1429		21.615		118.69		792.93
RDMeanB	B	-----	B	0.0006	B	0.0022	B	0.0105	B	0.0631	B	0.4779
REffB	B	-----	B	1.2727	B	1.2857	B	1.9514	B	2.7456	B	3.5972

By inspection of Table 7 it seems clear that for each pair (n, κ) average execution times computed for both the algorithms are quite close to their respective general average time $\bar{t}_n^{\mathrm{A}}$ and $\bar{t}_n^{\mathrm{B}}$. As for these latter, for all the n considered random variate generation by algorithm A brings with it general average times almost identical and included between 1.2 and 1.4 thousandth of second.

On the contrary, generation by algorithm B requires higher general average times, which grow quickly as n increase. In particular, this growth is not linear, as inconstant values of RDMeanB index reveal. For instance, from $n = 2$ to $n = 3$ generating the additional variable requires 0.6 thousandth of second on average more; from $n = 3$ to $n = 5$, 2.2 thousandth of second on average more are needed per variate, and so on, until 48 hundredth of second more per variate required for generating from 9 to 11 variables.

Moreover, by examining REffB index, we note that for three variates generating the time needed is nearly 1.27 times greater on average than for two variables is, while in five variates generation the time involved is about 1.29 times greater on average than for three variables is. Hence, relative efficiencies of algorithm B, respectively, for five variables compared with

three and for three variables compared with two are very close.

The reason why in these two situations relative efficiency of algorithm B is very similar is due to the shape of function $\mathcal{K}_n$ in (14). As remarked in Section 4, the implemented AR method is characterized by a different relative efficiency as n varies, measured by the inverse of $\mathcal{K}_n$, the expected number of trials per random variate (formula 14 and Figure 2). In this respect, we have already noted that function $\mathcal{K}_n$ takes its minimum value for $n=5$.

In the remaining pairwise comparisons REffB index is seen to increase quickly. For instance, as n increases from 9 to 11 variate generation takes about 3.60 times greater on average per additional variables.

Finally, by comparing Algorithm A and B directly in terms of relative efficiency (REff), in case of bivariate generation from spherical MEPs algorithm B is about 1.83 times slower than A; for three variates generating it is about 2.15 times slower than A, and so on until eleven variates generation, where algorithm B is about 793 times slower than A! This makes algorithm B rather impractical when the number of variables is greater than ten.

6. Conclusions

In this work, we propose a simple and fast procedure for generating from MEPs. These distributions belong to multivariate elliptical and symmetric Kotz family, so that with the purpose of generating by relating on known theoretical results a transformation approach to multivariate simulation is introduced.

On the other hand, following Devroye (1997) we consider a further procedure for generating from MEPs based on a general AR method. The two algorithms have been implemented in R language, vs. 1.8.1. We then compare them empirically by evaluating average times for routine execution.

As expected, the algorithm based on transformation approach is clearly more computationally efficient than the other one (Table 7). However, the AR algorithm here proposed is still in a naïve version; in order to fasten AR algorithms, methods that are just conceived for reduction of the number of rejections might be considered. Among these methods, the so-called Adaptive Table Methods lead to better rejection regions by employing knowledge from already generated data (Devroye, 1986).

Appendix A

A.1 Computation for Method A

- Proof of (10): $R \overset{\mathrm{d}}{=} \sqrt[\kappa]{\mathrm{Gam}\left(\frac{n}{\kappa}, \frac{1}{2}\right)}$, where R is the random variable in the stochastic representation (5) for $\boldsymbol{Y} \sim \mathrm{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$.

In this case, the density function generator $g(q)$, which appears in (3), is: $g(q) = C_{n,\kappa} \exp\left\{-\frac{1}{2} q^{\frac{\kappa}{2}}\right\}$, where q denotes the quadratic form $\|\mathbf{y}\|^2$ and $C_{n,\kappa} = n\Gamma\left(\frac{n}{2}\right) / \pi^{\frac{n}{2}} \Gamma\left(1 + \frac{n}{\kappa}\right) 2^{1+\frac{n}{\kappa}}$. Then, by (6) we obtain:

$$f_R(r) = \frac{\kappa}{2^{\frac{n}{\kappa}} \Gamma\left(\frac{n}{\kappa}\right)} r^{n-1} \mathrm{e}^{-\frac{1}{2} r^{\kappa}} . \tag{A1}$$

By simply changing $W = R^{\kappa}$ in (A1) we obtain:

$$\psi_W(w) = \frac{1}{2^{n/\kappa} \Gamma\left(\frac{n}{\kappa}\right)} w^{\frac{n}{\kappa} - 1} \exp\left\{-\frac{1}{2} w\right\}, \tag{A2}$$

that is the d.f. of a Gamma variable with shape parameter n/κ and scale parameter $1/2$. Expression (10) then follows.

A.2 Computation for Method B

- Proof of (12): $Y_M \overset{\mathrm{d}}{=} \sqrt[\kappa]{\mathrm{Gam}\left(\frac{n}{\kappa}, \frac{1}{2}\right)}$, where $Y_{\mathrm{M}} = \max\{Y_1, \ldots, Y_n\}$, with $Y_i \geq 0$ for all i.

For the case $\boldsymbol{Y} \sim \mathrm{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$, the density generator $f_{Y_M}(y_M)$, $y_M > 0$, of MMax distribution on positive quadrant is given by:

$$\begin{aligned} f_{Y_M}(y_M) &= c_n n y_M^{n-1} h(y_M) = c_n n y_M^{n-1} \frac{n\Gamma\left(\frac{n}{2}\right)}{\pi^{\frac{n}{2}} \Gamma\left(1 + \frac{n}{\kappa}\right) 2^{1+\frac{n}{\kappa}}} \exp\left\{-\frac{1}{2} y_M^{\kappa}\right\} = \\ &= \frac{\kappa}{2^{\frac{n}{\kappa}} \Gamma\left(\frac{n}{\kappa}\right)} y_M^{n-1} \exp\left\{-\frac{1}{2} y_M^{\kappa}\right\}, \qquad y_M \geq 0 \end{aligned} \tag{A3}$$

where the normalizing constant c_n is the inverse of $\mathcal{K}_n$ in (14).

By simply substituting $W = Y_M^{\kappa}$ in (A3) we obtain the d.f. of a Gamma variable with shape parameter n/κ and scale parameter $1/2$, as in the previous result (A2). Finally, expression (12) follows.

Appendix B

B.1 Implementation of Algorithm A in R language (Table 4, Section 4)

- Generation from the r.v. $\boldsymbol{U}^{(n)}$, uniformly distributed on the unit hypersphere surface in R^n

```
unif.gen <- function(n, N = 1){
      # n = number of variates #
      # N = number of values per random variate #
      x <- matrix(rnorm(n * N), N)
      norm.vec <- function(y){y/sqrt(sum(y^2))}
      unif <- t(apply(x, 1, norm.vec))
      unif
}
```

- Generation from $\boldsymbol{Y} \sim \text{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$

```
mep.gen.sph <- function(n, k, N = 1){
      # n = number of variates #
      # k = non normality parameter #
      # N = number of values per random variate #
      el.prod <- function(X){
      X <- as.matrix(X)
      matrix(X[,1] * c(X[, -1]), nrow(X))}
      x <- matrix(rgamma(N, n/k, 0.5)^(1/k), N)
      unif.vec <- do.call("unif.gen", list(n, N))
      ris.parz <- cbind(x, unif.vec)
      data.gen <- el.prod(ris.parz)
      data.gen
}
```

B.2 Implementation of Algorithm B in R language (Table 5, Section 4)

- Generation from the MMax distribution

```
max.gen <- function(n, k, N = 1){
      # n = number of variates #
```

```
      # k = non normality parameter #
      # N = number of values per random variate #
      el.prod <- function(X){
      X <- as.matrix(X)
      matrix(X[,1] * c(X[, -1]), nrow(X))}
      x <- matrix(rgamma(N, n/k, 0.5)^(1/k), N)
      vec.unif <- matrix(runif(n * N), N)
      rnd.int <-
      function(y,M){ceiling(matrix(runif(M,0,y),M))}
      indices <- cbind(1:N, rnd.int(n, N))
      vec.unif[indices] <- rep(1, N)
      vec.unif.new <- vec.unif
      ris.parz <- cbind(x, vec.unif.new)
      vec.fin <- el.prod(ris.parz)
      vec.fin
}
```

- Generation from $\boldsymbol{Y} \sim \mathrm{MEP}_n(\mathbf{0}, \mathbf{I}, \kappa)$

```
mep.AR.gen <- function(n, k, N = 1){
      # n = number of variates #
      # k = non normality parameter #
      # N = number of values per random variate #
      test.acc.rif <- function(n, k) {
      repeat {
            u <- runif(1)
            x <- do.call("max.gen", list(n, k))
            if (u*exp(-0.5*(max(x))^k) <=
                  exp(-0.5*(x%*%t(x))^(k/2)) )
            break
             }
      last.x <- x
      last.x}
      # Positive quadrant #
      mep.mtr <- numeric(N)
      mep.mtr <- eval(substitute(t(sapply(mep.mtr,
      function(test.acc.rif, n,k) test.acc.rif(n, k)))))
      # Random signs #
      signs <- matrix(floor(runif(N*n, min=-0.5,
               max=0.5)), N)
      signs[signs == 0] <- 1
      mep.mtr.new <- signs * mep.mtr
      mep.mtr.new
}
```

References

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

Devroye L. (1997). Random variate generation for multivariate unimodal densities. *ACM Transactions on Modeling and Computer Simulation*, **7**, 447-477.

Fang K.T., Kotz S., Ng K.W. (1990). *Symmetric Multivariate and Related Distributions.* Monographs on Statistics and Applied Probability, **36**, Chapman and Hall, New York.

Johnson M.E.(1987). *Multivariate Statistical Simulation.* Wiles Series in Probability and Mathematical Statistics, New York.

R Development Core Team (2003). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Rubinstein R.Y. (1981). *Simulation and the Monte Carlo method*. Wiles Series in Probability and Mathematical Statistics, New York.

Solaro N., Ferrari P. (2003a). Robustness of parameter estimation procedures in multilevel models. In: *Atti del Convegno SIS 2003*, Napoli.

Solaro N., Ferrari P. (2003b). On parameters estimation procedures in multilevel models. In: *Atti del Convegno SCO 2003*, Treviso.

Thisted R.A.(1988). *Elements of Statistical Computing.* Chapman and Hall, New York.