Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

ANALYSIS AND GENERATION OF RANDOM VECTORS WITH COPULAS

Johann Christoph Strelen

Rheinische Friedrich–Wilhelms–Universität Bonn
Römerstr. 164, 53117 Bonn, GERMANY

Feras Nassaj

Fraunhofer Institute for Applied Information Technology FIT
Schloss Birlinghoven
53754 Sankt Augustin, GERMANY

ABSTRACT

Copulas are used in finance and insurance for modeling stochastic dependency. They comprehend the entire dependence structure, not only the correlations. Here they are estimated from measured samples of random vectors. The copula and the marginal distributions of the vector elements define a multivariate distribution of the sample which can be used to generate random vectors with this distribution. This can be applied as well to time series. A programmed algorithm is proposed. It is fast and allows for random vectors with high dimension, for example 100.

1 INTRODUCTION

Stochastic models and discrete simulation are indispensable means for the quantitative analysis of systems. It is well known that missing to carefully model the influences from outside, especially the load, may lead to wrong results and ultimately to wrong decisions based on the simulation results. One reason for bad load models may be to ignore dependencies, i.e. to use independent random variables instead of proper commonly distributed random vectors or stochastic processes.

Influence from outside of the model like load or failure of system components can be incorporated into the model using observed traces or input models, namely random variables, random vectors, or stochastic processes. Data from traces can be used directly. If input is modeled, data are realisations of the model.

The use of random variates is well understood and common since long time, the use of generated random vectors and stochastic processes is much more difficult, not so popular, a topic of current research.

The use of copulas is common in finance and insurance. In this paper, we propose to use copulas for the analysis of observed data and for the generation of dependent random variates and time series.

The copula of a multivariate distribution describes its dependence structure completely, not only the correlations of the random variables. It is uncoupled from the marginal distributions which can be modeled as empirical distributions or fitted standard distributions as usual.

The use of copulas makes a difficult task, finding a multivariate distribution, more facile by performing two easier tasks. The first step is modeling the marginal distributions, the second consists in estimating the copula. Once we evaluated the estimated copula and marginal distributions, it is quite simple to use them to generate random vectors.

We model the marginal distributions as usual, and estimate the copula from a frequency distribution. This is not common, usually one of the known families of copulas is fitted to the sample. There are many such families, see e.g., Nelsen (1998), but the most for only two dimensions. For simulation, more dimensions might be needed. Moreover, as remarked in Blum, Dias, and Embrechts (2002), fitting a sample to a family of copulas is essentially as difficult as estimating the joint distribution in the first place. Thirdly, different families of copulas account for different kinds of dependence. Hence, the input modeler must choose the family according to the actual dependence nature. In contrast, an empirical copula incorporates the dependence form automatically. For these reasons, we use some kind of empirical copulas instead of fitting the samples to families of copulas.

The new technique contrasts with other proposed input models. Autoregressive processes (AR) model time series with Gaussian random variables. They are conveniently fitted to measured data with the linear Yule-Walker equations.

ARTA-like models (ARTA, Cario and Nelson 1996) for univariate time-series, NORTA (Cario and Nelson 1997) for random vectors, VARTA (Biller and Nelson 2003) for processes of random vectors) allow for general distributions by means of a Gaussian AR or a multivariate Gaussian random variable as basis whose random variables are transformed into the desired distributions. The correlations of the basis process are different from the desired correlations.

Sometimes this results in infeasible correlation matrices of the basis process (Ghosh and Henderson 2002), the *defective matrix problem*. Fitting given autocorrelations and distributions to an ARTA process is possible but not quite trivial.

TES processes (Melamed 1997) rely on empirical distributions of the random variables. They comprise lag 1 correlations. The interactive software system TEStool serves the purpose of fitting measured data to a TES process.

AR, ARTA-like, and TES processes as input modeling approaches for random vectors and time-series consider only the correlations, not the whole dependence structure. In contrast, copulas take into account the entire dependency, hence this new technique as well.

In Nassaj and J.Ch.Strelen (2005) we propose some kind of nonlinear non-Gaussian autoregressive processes. The dependence structure is more general, nonlinear dependencies are accounted for. The distributions of the random variables are general. The procedure of fitting to measured date is done in two successive steps. The first one for the dependence structure applies optimization with respect to some parameters. The second one concerns the distribution of univariate random variables. This separation is similar to the copula approach.

In Nassaj and J.Ch.Strelen (2006) we propose a method which is similar to the method of this paper. But there we apply heuristic approximations of the copula, here proper copulas.

1.1 Overview

We present a technique to efficiently generate random vectors with a distribution which is defined on the basis of a given sample of n random vectors which are realizations of a usually unknown D-dimensional distribution.

To this end we point out in Section 3 how to estimate marginal empirical distribution functions of the components in the form of step functions and an empirical copula from the sample.

In Section 4 we give an efficient algorithm for the generation of random D-variates with nearly the original D-dimensional distribution. An integer K controls the accuracy and the space and time complexity, all increasing with K. K may be in a range of about 10...4000.

Efficient means that the algorithm uses a moderate amount of storage of order O(D^2n), and provides fast generation in computing time of order O(D^2K) or even O(D^2 log K).

The generation algorithm performs in two steps:

1. Generate a random D-vector with the empirical copula.
2. Transform its elements with the inverses of the marginal distribution functions of the sample.

For Step 2 we propose three different alternatives:

- The estimated empirical distribution functions of the components in form of a step function. Here one can obtain only values which occur in the sample. This can be sensible for integer random variables, in particular.
- Some kind of linear interpolation. Here one can obtain values which lie between zero and the largest value of the sample, or something similar.
- Fitted standard distributions if feasible, where feasible means: The empirical distribution function of the sample is similar to a standard distribution function, and this standard distribution function can be inverted in a sufficiently simple manner.

We present examples in Section 5. They were calculated with a MATLAB program (Strelen 2007) and indicate accurate statistic properties of the new technique.

But first, in Section 2, we describe basic facts about copulas which are of interest for this article.

2 COPULAS

A compact definition of copulas is given in Pfeifer and Neslehova (2003):

Definition 1 *A* copula *is a function C of D variables on the unit D-cube $[0,1]^D$ with the following properties:*

1. The range of C is the unit interval $[0,1]$

2. $C(\mathbf{u})$ is zero for all $\mathbf{u}=(u_1,...,u_D)\in[0,1]^D$ for which at least one coordinate equals zero

3. $C(\mathbf{u})=u_d$ if all coordinates of $\mathbf{u}$ are 1 except the d-th one

4. C is D-increasing in the sense that for every $\mathbf{a}\le\mathbf{b}$ in $[0,1]^D$ the measure ΔC_a^b assigned by C to the D-box $[\mathbf{a},\mathbf{b}]=[a_1,b_1]\times...\times[a_D,b_D]$ is nonnegative, i.e.

$$\Delta C_a^b := \sum_{(\varepsilon_1,...,\varepsilon_D)\in\{0,1\}} (-1)^{\varepsilon_1+...+\varepsilon_D} C\Big(\varepsilon_1 a_1 + (1-\varepsilon_1)b_1,...,\varepsilon_D a_D+(1-\varepsilon_D)b_D\Big)\ge 0.$$

In fact, a copula is a multivariate distribution function for the random vector $\mathbf{U}=(U_1,...,U_D)$ with univariate uniform margins restricted to the unit D-cube. All partial derivatives exist almost everywhere, hence the conditioned distribution functions and the density as well.

The key theorem due to Sklar clarifies the relations of dependence and the copula of a distribution:

Theorem 1 (Sklar) *Let F denote a D-dimensional distribution function with margins $F_1,...,F_D$. Then there exists a D-copula C such that for all real $\mathbf{z}=(z_1,...,z_D)$,*

$$F(\mathbf{z})=C\Big(F_1(z_1),...,F_D(z_D)\Big).$$

If all the margins are continuous, then the copula is unique; in general, it is determined uniquely on the ranges of the marginal distribution functions. Moreover, if we denote by $F_1^{-1},...,F_D^{-1}$ quasi-inverses of the marginal distribution functions, then for every $\mathbf{u} = (u_1,...,u_D)$ in the unit D-cube,

$$C(\mathbf{u}) = F\left(F_1^{-1}(u_1),...,F_D^{-1}(u_D)\right).$$

For a proof, see Nelsen (1998). We define quasi-inverses of distribution functions as

$$F_d^{-1}(u) = \begin{cases} \inf\{z|F_d(z) \geq u\} & u > 0 \\ \sup\{z|F_d(z) = 0\} & u = 0 \end{cases}$$

Univariate random variates can be generated with the common inverse distribution function method. Similarly, multivariate random vectors can be generated using copulas in two steps:

1. Generate dependent random numbers $u_1^{(\text{gen})},...,u_D^{(\text{gen})}$ with the copula. This can be done in the usual way for the generation of random vectors, see Law and Kelton (2000), page 479), for example.

2. Transform them into the desired marginal distribution, $z_d^{(\text{gen})} = F_d^{-1}(u_d^{(\text{gen})})$, $d = 1,...,D$.

For step 1, we need some conditional distribution functions, namely $C_d(u_d|u_1,...,u_{d-1}) = P\{U_d \leq u_d|U_1 = u_1,...,U_{d-1} = u_{d-1}\}$, $d = 1,...,D$.

Let $c(\mathbf{u}) \equiv c_D(\mathbf{u})$ denote the density of the copula, and $c_d(\mathbf{u})$, $d = 1,...,D-1$, the needed marginal densities,

$$c_d(u_1,...,u_d) = \int_{u_{d+1}=0}^{1} ... \int_{u_D=0}^{1} c(\mathbf{u})du_{d+1}...du_D. \tag{1}$$

Then we have for $c_{d-1}(u_1,...,u_{d-1}) > 0$

$$C_d(u_d|u_1,...,u_{d-1}) = \frac{\int_{u=0}^{u_d} c_d(u_1,...,u_{d-1},u)du}{c_{d-1}(u_1,...,u_{d-1})}, d = 2,...,D. \tag{2}$$

A random variate of the copula is generated as follows. $u_1^{(\text{gen})}$ is a random variate from the uniform distribution on the interval $[0,1]$, denoted by $U(0,1)$, and $u_d^{(\text{gen})}$, $d = 2,...,D$, are generated with the conditional distribution functions $C_d(u_d|u_1^{(\text{gen})},...,u_{d-1}^{(\text{gen})})$, in turn.

3 PIECEWISE LINEAR EMPIRICAL COPULAS

We consider empirical copulas $C(\mathbf{u})$, $\mathbf{u} = (u_1,...,u_D)$, which are linear in each variable u_d if the other variables are fixed. This holds within certain subcubes of the unit D-cube $[0,1]^D$, but in different subcubes, the slopes are different, in general. The density $c(\mathbf{u})$ of the copula is constant in each of these subcubes. We call these copulas piecewise linear, more precisely piecewise multi-linear.

In this section, we point out how the copulas can be estimated from a given sample as well as the empirical distribution functions of the marginal distributions of the sample, we prove that they are really copulas, we present an algorithm for computing them and their conditional distribution functions which we need for generation, and we consider time and space complexity of this algorithm.

The subcubes are defined with the partition $S_1,...,S_K$ of the interval $[0,1]$, $S_1 = [0,\delta]$ where $\delta = 1/K$, K positive integer, and $S_j = \left((j-1)\delta, j\delta\right]$, $j = 2,...,K$. The subcubes are $\mathscr{S}_\mathbf{j} = S_{j_1} \times ... \times S_{j_D}$, $\mathbf{j} = (j_1,...,j_D) \in \mathscr{K} = \{1,...,K\}^D$.

We denote the given sample under consideration as $\mathbf{z}_i = (z_{1,i},...,z_{D,i})$, $i = 1,...,n$. n is the sample size. We assume i.i.d. sample vectors $\mathbf{z}_i$.

For the definition of the copula, we use in each dimension $d = 1,...,D$ the order statistics $z_{d,(i)}$, $i = 1,...,n$, of the elements of dimension d. Let $I_d : \{1:n\} \to \{1:n\}$ denote the function which maps the old place l of $z_{d,l}$ in the sample to the new place $I_d(l) = i$ in the ordered sequence.

With this mapping, we define $u_{d,l} = I_d(l)/n$, $l = 1,...,n$, $d = 1,...,D$, and $\mathbf{u}_l = (u_{1,l},...,u_{D,l})$, $l = 1,...,n$. $\mathbf{u}_l$ are image points of the sample points $\mathbf{z}_l$ in the unit D-cube $[0,1]^D$.

Now we explain that the values $u_{d,l}$ are closely related to the empirical distribution function of the $z_{d,l}$. We use a specific kind of empirical distribution functions, namely step functions, one in each dimension $d = 1,...,D$. They are defined with the order statistics as follows:

$$F_d(z) = \begin{cases} 0 & z < z_{d,(1)} \\ i/n & z_{d,(i)} \leq z < z_{d,(i+1)},\ i = 1,...,n-1 \\ 1 & z_{d,(n)} \leq z \end{cases} \tag{3}$$

Remark 1. If in the ordered sequence $z_{d,(i)} = z_{d,(i+1)}$, then there is no z for which $F_d(z) = i/n$, i/n is not in the range of F_d.

Remark 2. If all sample values $z_{d,l}$, $l = 1,...,n$, are different, $F_d(z_{d,l}) = u_{d,l}$ holds. If not, this is true only for values $z_{d,l}$ which occur only once, and for just one of some equal values. Namely for multiple values $z_{d,l_1} = z_{d,l_2} = ...$, $F_d(z_{d,l_i}) = \max\{u_{d,l_1}, u_{d,l_2}, ...\}$ holds for $i = 1,2,...$.

Let $N_\mathbf{j} = |\{\mathbf{u}_i|\mathbf{u}_i \in \mathscr{S}_\mathbf{j},\ i = 1,...,n\}|$ denote the number of points in $\mathscr{S}_\mathbf{j}$, $\mathbf{j} \in \mathscr{K}$. Then we define the density $c(\mathbf{u}) \equiv c_D(\mathbf{u})$ of the copula as

$$c(\mathbf{u}) = f_\mathbf{j},\ \mathbf{j} \in \mathscr{K},$$

where we use the notation $f_\mathbf{j} = N_\mathbf{j}/n/\delta^D$.

For $u_d \in S_j \setminus \{0\}$, $j = \lceil u_d K \rceil$ holds, and for $u_d = 0$, $j = 1$. Hence, with the up-operator $\uparrow : [0,1] \to \{1,...,K\}$, $\uparrow u$

$= \max\{1, \lceil uK \rceil\}$, we can write

$$c(\mathbf{u}) = f_{\uparrow u_1,\dots,\uparrow u_D}, \ \mathbf{u} \in [0,1]^D. \tag{4}$$

The marginal densities (1) can be expressed as

$$c_d(u_1,\dots,u_d) = \delta^{D-d} f^{(d)}_{\uparrow u_1,\dots,\uparrow u_d}, \ d = 1,\dots,D-1, \tag{5}$$

where

$$f^{(d)}_{j_1,\dots,j_d} = \sum_{(j_{d+1},\dots,j_D)=(1,\dots,1)}^{K\times\dots\times K} f_{\mathbf{j}},$$
$$(j_1,\dots,j_d) \in \{1,\dots,K\}^d,$$

and the distribution functions (2) as

$$\begin{aligned}
&C_2(u_2|u_1) = \\
&\quad \delta^{D-1} \sum_{j_2=1}^{\downarrow u_2} f^{(2)}_{\uparrow u_1, j_2} + (u_2 - \downarrow u_2 \delta)\delta^{D-2} f^{(2)}_{\uparrow u_1, \uparrow u_2}, \\
&C_d(u_d|u_1,\dots,u_{d-1}) = \\
&\quad \frac{\delta \sum_{j_d=1}^{\downarrow u_d} f^{(d)}_{\uparrow u_1,\dots,\uparrow u_{d-1}, j_d} + (u_d - \downarrow u_d \delta) f^{(d)}_{\uparrow u_1,\dots,\uparrow u_d}}{\delta f^{(d-1)}_{\uparrow u_1,\dots,\uparrow u_{d-1}}}, \\
&\quad d = 3,\dots,D,
\end{aligned} \tag{6}$$

with the down-operator $\downarrow$: $[0,1] \rightarrow \{1,\dots,K\}$, $\downarrow u = \uparrow u - 1$, for positive denominators.

Now we are prepared to prove

Theorem 2 *If the sample size n and K are such that K divides n, $C(\mathbf{u})$ is a copula.*

Proof It suffices to prove that $c(\mathbf{u})$ is a density in the D-cube $[0,1]^D$, and that $c_1(u)$ is the uniform density over $[0,1]$.

From (4), we conclude $c(\mathbf{u}) \geq 0$. Integrating the $c(\mathbf{u})$ over the D-cube, one obtaines $\delta^D \sum_{\mathbf{j}\in\mathscr{K}} f_{\mathbf{j}} = \delta^D n/n/\delta^D = 1$. Hence $c(\mathbf{u})$ is a density.

From the definition of $I_d(l)$, one can conclude that for fixed $d \in \{1,\dots,D\}$, $I_d(l)$ for all $l = 1,\dots,n$ assumes all values $i = 1,\dots,n$, each value once. Hence, the $u_{d,l}$ for all $l = 1,\dots,n$ assume all values $1/n,\dots,(n-1)/n, 1$, each value once. Hence, in each subcube $S_{j_1} \times [0,1]^{D-1}$ are n/K image points $\mathbf{u}_l$. From (5) we find $f^{(1)}_{j_1} = n/K/n/\delta^D = 1/\delta^{D-1}$ and $c_1(u_1) = 1$. $c_1(u)$ is the uniform density over $[0,1]$. This concludes the proof.

The following algorithm can be applied to calculate the arrays $f^{(d)}$ which we use for the calculation of the conditional distribution functions of the copula.

Algorithm 1

```
1.  Calculate the empirical distribution functions of the
    marginal distributions of the sample with (3) and the
    image points $\mathbf{u}_i$ of the sample points $\mathbf{z}_i$, $i = 1,...,n$.
2.  Set all elements of the arrays $f^{(d)}$ to 0.
3.  for $i := 1$ to $n$ do
        for $d := 1$ to $D$ do
            $j_d := \uparrow u_{d,i}$;
        end
        for $d := 2$ to $D$ do
            $f^{(d)}_{j_1,...,j_d}$ plus $1/n/\delta^D$;
        end
    end
```

Each of the D empirical distributions need $\mathrm{O}(n \log n)$ steps for sorting n numbers, so this is done in $\mathrm{O}(Dn \log n)$ time. The loops consist in only $\mathrm{O}(Dn)$ steps.

For the space complexity, we remark that the arrays can be expected to be sparse. The biggest one, $f^{(D)}$, has K^D elements, most of which will be zero if $K^D \gg n$. Therefore we propose to store the big arrays $f^{(d)}$ in a sparse manner where elements with value zero are not stored. Hence, if K^D is not small, the needed storage is of order $O(Dn)$ since at most $(D-1)n$ times an array element changes its value from zero to nonzero. So we conclude the space complexity $O(Dn)$.

But, when an array is stored in a sparse manner, the index values must be stored together with each array element, and each time when an element is accessed, its index values are read. This results in a space and time complexity $O(D^2 n)$

So we can state

Theorem 3 *Algorithm 1 has the time complexity $O(D^2 n \log n)$, and the data structures have the space complexity $O(D^2 n)$.*

4 THE GENERATION ALGORITHM

The generation algorithm consists in two steps. First a random D-vector $\mathbf{u}^{(\mathrm{gen})}$ is generated with the copula as indicated at the end of Section 2. Then each of its elements u_d, $d = 1,\dots,D$, of $\mathbf{u}^{(\mathrm{gen})}$ is transformed with the inverse of the according marginal distribution function of the sample (The superscript $^{(\mathrm{gen})}$ would overload some formulas of this section, therefore we omit it at the elements of $\mathbf{u}^{(\mathrm{gen})} = (u_1,\dots,u_D)$).

In step 1, u_1 is a random variate from the uniform distribution on the interval $(0,1]$, denoted by $U(0,1]$ (We exclude the random number value zero; this makes the implementation of the quasi-inverse of distribution functions easier). Then u_d, $d = 2,\dots,D$, are generated with the conditional distribution functions (6). To this end, for each $d = 2,\dots,D$, a random number u is generated from the distribution $U(0,1]$, and $u = C_d(u_d|u_1,...,u_{d-1})$ is solved for u_d.

For the u_d we obtain the following formulas:

$$u_2 = \downarrow u_2 \delta + \frac{u - \delta^{D-1}\sum_{j_2=1}^{\downarrow u_2} f^{(2)}_{\uparrow u_1, j_2}}{\delta^{D-2} f^{(2)}_{\uparrow u_1, \uparrow u_2}} \quad (7)$$

where $\downarrow u_2$ is the smallest integer in $\{0,...,K-1\}$ for which

$$uK^{D-1} \leq \sum_{j_2=1}^{\downarrow u_2+1} f^{(2)}_{\uparrow u_1, j_2} \quad (8)$$

holds. For $d = 3,...,D$, the formula is

$$u_d = \downarrow u_d \delta + \delta \frac{u f^{(d-1)}_{\uparrow u_1,...,\uparrow u_{d-1}} - \sum_{j_d=1}^{\downarrow u_d} f^{(d)}_{\uparrow u_1,...,\uparrow u_{d-1}, j_d}}{f^{(d)}_{\uparrow u_1,...,\uparrow u_d}} \quad (9)$$

where $\downarrow u_d$ is the smallest integer in $\{0,...,K-1\}$ for which

$$u f^{(d-1)}_{\uparrow u_1,...,\uparrow u_{d-1}} \leq \sum_{j_d=1}^{\downarrow u_d+1} f^{(d)}_{\uparrow u_1,...,\uparrow u_{d-1}, j_d} \quad (10)$$

holds.

If the random vector $\mathbf{u}^{(\text{gen})}$ is generated with these formulas, the calculation of the sums in (7) and (9) consist in $O(DK)$ steps for calculating the sum with sparse storage organization; the same holds for the search of $\downarrow u_d$ with (8) and (10), for each dimension d. Hence this needs computing time of order $O(D^2K)$.

This can be reduced to $O(D^2 \log K)$ as follows, which is worthwile for large K. For example, if K=1000, the number of steps is about $10D$ instead of $500D$, in average. In a setup phase of the generation algorithm, the cumulative sums

$$s^{(d)}_{j_1,...,j_d} = \sum_{j=1}^{j_d} f^{(d)}_{j_1,...,j_{d-1},j}, \quad (11)$$

$$(j_1,...,j_{d-1}) \in \{1,...,K\}^{d-1}, \quad (12)$$

$$j_d = 0,...,K, \ d = 2,...,D, \quad (13)$$

are calculated. In the formulas, the sums are replaced by a single element of the array s, and the $\downarrow u_d$ are determined with binary search. So we found

Theorem 4 *A random vector* $\mathbf{u}^{(gen)}$ *can be generated in* $O(D^2 \log K)$ *time.*

It must be remarked that the price for the logarithmic time complexity is a high space need for the s array. But in the next subsection we propose a tailored data structure for storing the f- and s-arrays which realizes the favourable space complexity $O(D^2 n)$ and provides fast access.

Now we discuss the transformation of a random vector $\mathbf{u}^{(\text{gen})}$ into a random vector $\mathbf{z}^{(\text{gen})}$ with the original distribution as defined by the sample.

For the transformation, the elements of $\mathbf{u}^{(\text{gen})}$ are transformed with the inverses of the marginal distribution functions of the sample. We propose three different approaches:

1. Using the estimated empirical distribution functions (3). Here one can obtain only values which occur in the sample. This can be sensible for integer random variables, in particular.
2. Using a linear interpolation. The flat steps of the step distribution function are replaced by straight lines above them with positive slope. Here one can obtain values which lie between zero and the largest value of the sample - different highest and lowest values can be defined similarly.
3. Using fitted standard distributions if feasible, where feasible means: The empirical marginal distribution (3) of the sample is similar to a standard distribution function, and this standard distribution function can be inverted in a sufficiently simple manner.

Method 1 The elements of $\mathbf{z}^{(\text{gen})}$ are defined by

$$z^{(\text{gen})}_{d,l} = z_{d,(i)}, \ l = 1,...,n, \ d = 1,...,D, \quad (14)$$

where $i = \max\{1, \lceil u_{d,l} n \rceil\}$.

Method 2 The elements of $\mathbf{z}^{(\text{gen})}$ are defined by

$$z^{(\text{gen})}_{d,l} = \begin{cases} u_{d,l} n z_{d,(1)} & i = 0 \\ z_{d,(i)} + (u_{d,l} n - i)(z_{d,(i+1)} - z_{d,(i)}) & i > 0 \end{cases}$$
$$\text{where } \ i = \max\{1, \lceil u_{d,l} n \rceil\} - 1,$$
$$l = 1,...,n, \ d = 1,...,D. \quad (15)$$

Method 3 For a fixed $d \in \{1,...,D\}$, the sample $z_{d,1},...,z_{d,D}$ is fitted to a suitable standard distribution with invertible distribution function, say $F^{(\text{standard})}$, and the random variates are transformed as follows:

$$z^{(\text{gen})}_{d,l} = F^{(\text{standard})-1}(u_{d,l}), \ l = 1,...,n. \quad (16)$$

4.1 A Tailored Data Structure

When arrays or even sparse matrices are used for storing the array elements $f^{(d)}_{j_1,...,j_d}$ and their cumulative sums $s^{(d)}_{j_1,...,j_d}$, the applicability of our method is restricted with respect to the precision number K and the dimension D of the random vectors. Something like $K \leq 1000$ and $D \leq 3$, or

$K \leq 30$ and $D \leq 6$ must be observed with full arrays, and with sparse matrices in MATLAB, $K \leq 1000$ and $D \leq 6$, or $K \leq 30$ and $D \leq 12$. Therefore we devised a hash-based data structure which realizes the more favourable space and time complexity and therefore makes bigger K and D possible. We tried our program with $K = 4000$ and $D = 4$, with $K = 1000$ and $D = 40$, and with $K = 100$ and $D = 100$, for example. Moreover, the algorithm became much faster. For big values K, we observed 30...300 times shorter CPU times for the setup phase and 24...90 times faster generation of random vectors, compared with a MATLAB program which relies on sparse matrices.

We only sketch the basic ideas of the data structure which includes many details and is quite tricky.

In a first phase, the $f^{(d)}$-values are included one after the other into a hash table. The hash address depends on d and on the index tuple $(j_1, ..., j_{d-1})$.

In a second phase, the data structure is reorganized to allocate sequentially the cumulative sums $s^{(d)}_{j_1,...,j_d}$ for given $(j_1, ..., j_{d-1})$. This allows for binary search as will be seen in the sequel.

After that, the array elements $f^{(d)}_{j_1,...,j_d}$ and $s^{(d)}_{j_1,...,j_d}$ are accessed as follows.

1. Given d and $(j_1, ..., j_d)$, an entry is searched in the hash table. If none is found, the $f^{(d)}$-element or the $s^{(d)}$-element is 0. Otherwise, in the entry are two pointers, $begin(d, j_1, ..., j_{d-1})$ and $end(d, j_1, ..., j_{d-1})$ which point to triples $(j_d, f^{(d)}_{j_1,...,j_d}, s^{(d)}_{j_1,...,j_d})$ in a list.
2. In the list, between the pointers, all positive f-values for the given d and $(j_1, ..., j_{d-1})$ are stored. The triples are sorted according to increasing j_d which includes increasing cumulative sums $s^{(d)}_{j_1,...,j_d}$. If there is no triple for the given j_d-value, the array element $f^{(d)}_{j_1,...,j_d}$ is 0.

When the triples are searched linearly between the pointers, the access time is $O(K)$ plus the access time to find the pointers in the hash table which is $O(D)$.

With binary search, the access time is only $O(\log K) + O(D)$.

Due to the sparsity which is often high-grade, there are only a couple of $f^{(d)}_{j_1,...,j_d} > 0$, given d and $(j_1, ..., j_{d-1})$. Therefore quite often binary searching the triples is not better, but sequential search is faster. Therefore we use sequential search in the MATLAB program pwlCopula_D_with_hashing (Strelen 2007).

4.2 Time Series

The technique for random vectors can be applied for time series as follows. Consider a stationary time series $\mathbf{t}_i$, $i = 1, ..., n+m-1$, of D'-dimensional random vectors, $\mathbf{t}_i = (t_{1,i}, ..., t_{D',i})$. A moving window with m vectors $\mathbf{t}_{i-m+1}, \mathbf{t}_{i-m+2}, ..., \mathbf{t}_i$ is taken as sample vectors $\mathbf{z}_i = (t_{i-m+1,1}, ..., t_{i-m+1,D'}, t_{i-m+2,1}, ..., t_{i-m+2,D'}, t_{i,1}, ..., t_{i,D'})$, $i = m, ..., n+m-1$, hence $D = mD'$ is the dimensionality of the random vectors $\mathbf{z}_i$. With this sample, the marginal distributions and the copula are estimated as in Section 3. The reader may realize that there are only D' different marginal distributions.

The generation of a time series $\mathbf{t}^{(\text{gen})}_i, i = 1, 2, ...,$ is different. In each generation step i, only the last D' elements are newly generated, and not the whole vector $\mathbf{z}^{(\text{gen})}_i$. The first $(m-1)D'$ elements for the new $\mathbf{z}^{(\text{gen})}_i$ are taken from $\mathbf{z}^{(\text{gen})}_{i-1}$ instead, namely its last $(m-1)D'$ elements. Only the last D' elements are generated newly each time. The same holds for the $\mathbf{u}^{(\text{gen})}_i$-vectors.

The first vector $\mathbf{z}^{(\text{gen})}_1$ must be initialized somehow, since no older random vector is available. For this purpose, the whole vector can be generated.

The last D' elements of the generated $\mathbf{z}^{(\text{gen})}_i$ series are the desired generated time series $\mathbf{t}^{(\text{gen})}_i$, $i = 1, 2, ...$

It must be remarked that for this generation method of time series, two subtle problems must be solved:

1. All parts $\mathbf{t}_j$ of the vectors $\mathbf{z}_i$ must have the same empirical marginal distributions.
2. For each $\mathbf{t}_i$ in $\mathbf{z}_i = (..., \mathbf{t}_i)$, there must be other vectors $\mathbf{z}_j$ with this $\mathbf{t}_i$ in the lower places, like $\mathbf{z}_j = (\mathbf{t}_i, ...)$, $\mathbf{z}_j = (..., \mathbf{t}_i, ...)$.

Both postulations are not immediately true and must be forced explicitly. If they are not fulfilled, the generation algorithm may run into dead ends. In our MATLAB program pwcCopula_D_with_hashing (Strelen 2007), this problem is solved. We omit here the messy details.

5 EXAMPLES

We present numerical examples where we applied our new technique. In all but the second example, random vectors $\mathbf{z}_i$ are generated as samples with known distributions and dependence structure. The second consists of measured IP traffic data.

We determined empirical marginal distributions and the copula with the methods of Section 3 and generated random vectors $\mathbf{z}^{(\text{gen})}_i$ as described in Section 4. We verified the accuracy with some statistics and, visually, with scatter diagrams. Scatter diagrams serve the purpose to judge if there are regions in in the D-dimensional space in which no sample points are present, and if the same holds for the generated random vectors. Moreover, they give some visual impression of the frequency distribution of sample points and generated points.

Example 1: A multivariate distribution with dimension $D = 2$. The $Z_{1,i}$ have a Weibull(3,1)-distribution, and $Z_{2,i} = Z_{1,i}(1+Y_i)$, $i = 1,...,n$. All $Z_{1,i}$ and Y_i are independent random variables, and the Y_i are $U(0,1)$-distributed.

Example 2: Measured IP-data. a_i and b_i are inter-arrival times and packet sizes, respectively. We consider data from Klemm, Lindemann, and Lohmann (2002). The sample vectors are $z_{1,i} = a_i$, $z_{2,i} = b_i$, $z_{3,i} = a_{i+1}$, $z_{4,i} = b_{i+1}$, $i = 1,...,n-1$, and $z_{1,n} = a_n$, $z_{2,n} = b_n$, $z_{3,n} = a_1$, $z_{4,n} = b_1$, $i = 1,...,n-1$. The values of $z_{3,n}$ and $z_{4,n}$ are a little strange; this setting is according to the remark 2 at the end of subsection time series.

Example 3: We consider the stochastic process where A_1 is $U(0,1)$-distributed, $A_{k+1} = 0.5(1-4(A_k-0.5)^2)+0.5Y_k$, $k = 2,...,n0+n$, and the sample vectors are $Z_{1,i} = A_{i+n0}$, $Z_{2,i} = A_{i+n0+1}$, $i = 1,...,n-1$, and $Z_{1,n} = A_{n+n0}$, $Z_{2,n} = Z_{1,1}$. The first random variables A_1, A_2, ... are not stationary; this is why we skip $n0 > 0$ realizations, actually $n0 = 100$. We hope that the stochastic process is then nearly stationary. Here, the dimension is $D = 2$.

Example 4: Here we consider random vectors with high dimensions D, namely $D = 5, D = 40$, and $D = 100$. $Z_{1,i} = Y_{1,i}$, $Z_{d,i} = Z_{\lceil d/2 \rceil,i}(1-Z_{\lceil d/2 \rceil,i}) + Y_{d,i}$, $d = 2,...,D$, $i = 1,...,n$, and the $Y_{d,i}$ are independent and $U(0,1)$-distributed.

Statistics and diagrams were calculated for both, the measured sample $\mathbf{z}_i$ and the random vectors $\mathbf{z}_i^{(\text{gen})}$ which are generated with the new technique.

The statistics are means, coefficients of variation, and correlations of the $z_{d,i}$, the latter between $z_{d,i}$ and $z_{d',i}$, $d \neq d'$, and correspondingly of the generated random vectors.

We calculated the relative differences of measured and generated means and differences of measured and generated coefficients of variation and correlations.

In order to not bother the reader with many figures, we only give the maximum of the absolute values of these differences, the *maximum statistics difference*.

Each generation of random vectors was repeated independently six times. We present the interval of the observed maximum statistics differences.

In many examples, we found that the generated random vectors seem to have very similar statistical properties, compared to the samples.

The examples were calculated with the MATLAB program pwcCopula_D_with_hashing, see Strelen (2007). The program can be adapted easily to different samples.

Example 1 Continued: Sample size $n = 4000$, $K = 1000$. The $\mathbf{u}_i^{(\text{gen})}$-vectors were generated with the formulas (7) and (8). As inverse transformation we applied method 2 with the formula (15), the interpolated empirical distribution function, and method 3 with the formula (16). For method 3, the empirical marginal distributions were fitted to Weibull distributions. We obtained Weibull(3.06,0.983) for dimension 1 and Weibull(2.65,1.48) for dimension 2, respectively.

For 16000 generated random vectors, the maximum statistics differences are 0.003 ± 0.002 and 0.022 ± 0.003, respectively, obviously better when using the inverse transformation method 2.

The scatter diagrams of the sample and the generated points are nearly equal for both inverse transformation methods. They indicate that there are obviously regions where no points can be, and that these regions are observed by the generated points quite accurately; see Figure 1.

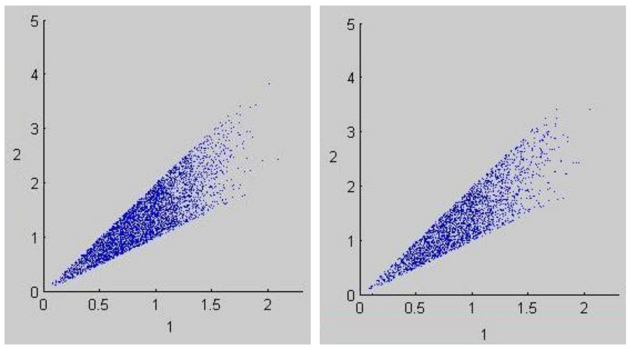

Figure 1: Sample and generated points.

Example 2 Continued: Sample size $n = 4000$. The $\mathbf{u}_i^{(\text{gen})}$-vectors were generated with formulas (7) - (10). As inverse transformation we applied method 1 with the formula (14), the empirical distribution function, for the integer-valued packet sizes, and Method 2 with the formula (15), the interpolated empirical distribution function, for the interarrival times.

For the accuracy $K = 1000$, in 4.8 seconds CPU time on a Pentium 4 with 3 GHz 16000 random vectors were generated with maximum statistics differences 0.028 ± 0.016. For the accuracy $K = 4000$, in 5.4 seconds CPU time 16000 random vectors were generated with maximum statistics differences 0.0035 ± 0.0005 which is obviously much more accurate. The set-up time of 2.6 seconds did not differ because it depends mainly on the dimension D and the sample size n.

The scatter diagrams show a very irregular dependency structure. The comparison of the sample and the generated points indicate good accuracy; see Figures 2, 3 and 4.

Example 3 Continued: Sample size $n = 4000$. For the accuracy $K = 1000$, in 5.1 seconds CPU time 64000 random vectors were generated with maximum statistics differences 0.0024 ± 0.0012. For the accuracy $K = 4000$, in 7.8 seconds CPU time 64000 random vectors were generated with maximum statistics differences 0.0002 ± 0.00002, very accurate.

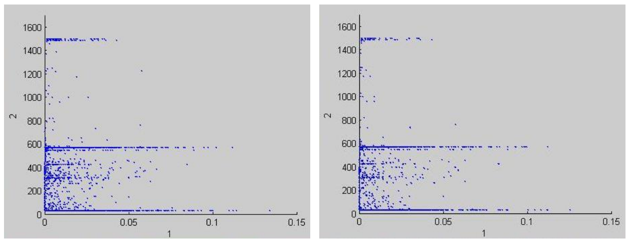

Figure 2: Sample and generated points, Dimensions 1 and 2.

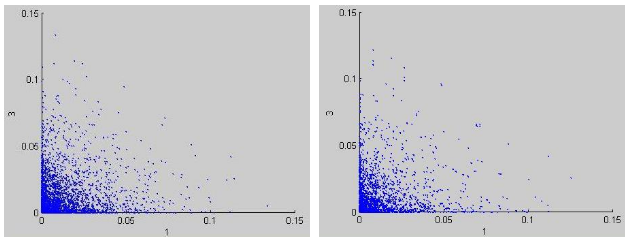

Figure 3: Sample and generated points, Dimensions 1 and 3.

Example 4 Continued: With sample size $n = 4000$, dimension $D = 5$, and accuracy $K = 4000$, in 29 seconds CPU time 64000 random vectors were generated with maximum statistics differences 0.007 ± 0.002. With sample size $n = 1000$, dimension $D = 40$, and accuracy $K = 1000$, in 74 seconds CPU time 16000 random vectors were generated with maximum statistics differences 0.028 ± 0.004. With sample size $n = 1000$, dimension $D = 100$, and accuracy $K = 100$, in 210 seconds CPU time 16000 random vectors were generated with maximum statistics differences 0.031 $\pm$ 0.0035.

CONCLUSION

Copulas seem to be useful for the analysis of multivariate samples and for the generation of multivariate random vectors and time series. Probably the proposed technique is a little step towards the consideration of dependency in load models where needed. The cited computer program should be developed further to become a useful tool.

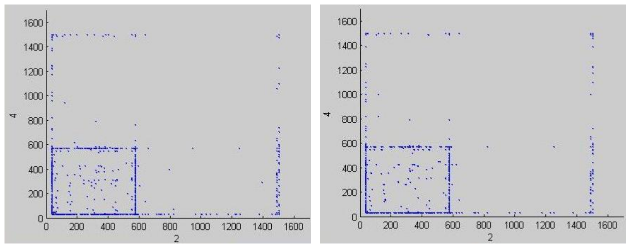

Figure 4: Sample and generated points, Dimensions 2 and 4.

REFERENCES

Biller, B., and B. Nelson. 2003. Modeling and generating multivariate time-series input processes using a vector autoregressive technique. *ACM Transactions on Modeling and Computer Simulation* 13:211–237.

Blum, P., A. Dias, and P. Embrechts. 2002. The art of dependence modelling: the latest advances in correlation analysis. 339–356. London.

Cario, M., and B. Nelson. 1996. Autoregressive to anything: Time-series input processes for simulation. *Operations Research Letters* 19:51–58.

Cario, M., and B. Nelson. 1997. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Tech. rep., Northwestern University, Department of Industrial Engineering and Management Sciences, Evanston, Ill.

Ghosh, S., and S. Henderson. 2002. Properties of the NORTA method in higher dimensions. In *Winter Simulation Conference Proceedings*, ed. E. Yücesan, C. Chen, J. Snowdon, and J. Charnes, 263–269. Piscataway, N.J.: IEEE.

Klemm, A., C. Lindemann, and M. Lohmann. 2002. Traffic modeling of IP networks using the batch Markovian arrival process. In *Proceedings of the 12th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, 92–110. London: Springer. LNCS 2324.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*. third ed. New York: McGraw-Hill.

Melamed, B. 1997. The empirical TES methodology: Modeling empirical time series. *J. of Applied Mathematics and Stochastic Analysis* 10 (4): 333–353.

Nassaj, F., and J.Ch.Strelen. 2005. Dependence input modeling with the help of non-Gaussian AR models and genetic algorithms. In *Modelling and Simulation 2005, Proceedings of the European Simulation and Modelling Conference, Porto, 2005*, ed. J. Teixera and A.E.Carvalho-Brito, 146–153. Ghent: Eurosis-ETI.

Nassaj, F., and J.Ch.Strelen. 2006. Generating simulation input with approximate copulas. In *Modelling and Simulation 2006, Proceedings of the European Simulation and Modelling Conference, Toulouse, 2006*, ed. A. Nketsa, M. Paludetto, and C. Bertelle, 88–93. Ghent: Eurosis-ETI.

Nelsen, R. 1998. *An introduction to copulas*. New York: Springer.

Pfeifer, D., and J. Neslehova. 2003. Modeling dependence in finance and insurance: the copula approach. *Blätter der DGVFM* 26 (2): 177–191. http://www.mathematik.uni-oldenburg.de/ personen/pfeifer/Copulas.pdf.

Strelen, J. C. 2007. Generating random vectors with copulas - MATLAB program pwlCopula_D_with_hashing. ⟨web.cs.uni-bonn.de/IV/strelen/Algorithmen⟩.

AUTHOR BIOGRAPHIES

JOHANN CHRISTOPH STRELEN received the Dipl.-Math. and Dr. rer. nat. degrees in mathematics and the Habilitation degree in Computer Science from the Technische Hochschule Darmstadt, Germany. He is Professor of Computer Science at the University of Bonn, Germany. His research interests include performance evaluation, distributed systems, and simulation. His e-mail address is ⟨strelen@cs.uni-bonn.de⟩ and his web address is ⟨web.cs.uni-bonn.de/IV/strelen/strelen_en.html⟩.

FERAS NASSAJ received his Bachelor degree in Electrical Engineering from the Aleppo University, Syria, in 2000, and his Master degree in Computational Engineering from the Friedrich-Alexander-University Erlangen-Nuernberg, Erlangen, Germany, in 2003. He is a scientific assistant at the Fraunhofer Institute in Sankt Augustin in Germany. His research interests include computer simulation and input modeling.