Communications in Statistics - Simulation and Computation

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/lssp20

Conditional Distribution Inverse Method in Generating Uniform Random Vectors Over a Simplex

A. Moeini [a] , B. Abbasi [b] & H. Mahlooji [c]

[a] Department of Industrial Engineering , Shahed University , Tehran, Iran

[b] School of Mathematical and Geospatial Sciences , RMIT University , Melbourne, Victoria, Australia

[c] Department of Industrial Engineering , Sharif University of Technology , Tehran, Iran

Published online: 16 Mar 2011.

To cite this article: A. Moeini , B. Abbasi & H. Mahlooji (2011) Conditional Distribution Inverse Method in Generating Uniform Random Vectors Over a Simplex, Communications in Statistics - Simulation and Computation, 40:5, 685-693

To link to this article: http://dx.doi.org/10.1080/03610918.2010.551012

Communications in Statistics—Simulation and Computation®, 40: 685–693, 2011

ISSN: 0361-0918 print/1532-4141 online
DOI: 10.1080/03610918.2010.551012

Conditional Distribution Inverse Method in Generating Uniform Random Vectors Over a Simplex

A. MOEINI[1], B. ABBASI[2], AND H. MAHLOOJI[3]

[1]Department of Industrial Engineering, Shahed University, Tehran, Iran
[2]School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Victoria, Australia
[3]Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran

Motivated by numerous applications in Monte Carlo techniques and as of late, in deriving non dominated solutions in multi-objective optimization problems, this article addresses generating uniform random variables $(\lambda_i, \lambda_i \geq 0, i = 1, \ldots, n)$ *over a simplex in* $\mathbb{R}^2$ $(n \geq 2)$*, i.e.,* $\sum_{i=1}^{n} \lambda_i = 1$*. In this article, first, conditional distribution of* λ_i *where* $\sum_{i=1}^{n} \lambda_i = 1$ *is derived and then inverse method is applied to generate random variables.*

Keywords Conditional probability distribution; Simplex; Uniform random vector generation.

Mathematics Subject Classification 65C10; 68Q25; 60E05.

1. Introduction

This article proposes a method to generate random numbers over a simplex. Recall that a simplex is an iterated pyramid, that is, a polytope whose number of vertices exceeds its dimension by one, such as a triangle and the standard pyramid.

Generating uniform random numbers over a simplex is required in wide array of simulation techniques. For instance, Pareto solution method in multi-objective optimization needs to assign different objective weights to each objective function so that it can generate multiple solutions. These weights are uniformly distributed and their sum is equal to 1 (Vetschera et al., 2010).

Received June 8, 2010; Accepted December 20, 2010
Address correspondence to B. Abbasi, School of Mathematical and Geospatial Sciences, RMIT University, L9, Building 8, Melbourne, Victoria 3001, Australia; E-mail: babak.abbasi@rmit.edu.au

A unit standard n-dimensional simplex in the n-dimensional Euclidean space $\mathbb{R}^n$ is defined as

$$\Omega_{\mathbf{n}} = \left\{ \lambda = (\lambda_1, \ldots, \lambda_n) : \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1 \right\}$$

If we generate data over $\Omega_{\mathbf{n}}$, then we can transform such data to spread over any simplex.

There are two efficient algorithms for uniform vector generation over a simplex introduced by Rubinstein (1982). The steps of these algorithms are outlined as follows.

Algorithm 1

Step 1. Generate $n-1$ independent uniform random numbers $U_1, \ldots, U_{n-1}$ from *Uniform*(0, 1).

Step 2. Arrange these numbers in ascending order to represent order statistics denoted by $U_{(1)}, \ldots, U_{(n-1)}$.

Step 3. Apply the following formulae and deliver $\lambda = (\lambda_1, \ldots, \lambda_n)$ as a vector distributed uniformly on the $\Omega_{\mathbf{n}}$:

$$\lambda_1 = U_{(1)}, \quad \lambda_2 = U_{(2)} - U_{(1)}, \ldots, \lambda_{n-1} = U_{(n-1)} - U_{(n-2)}, \quad \lambda_n = U_{(n-1)}.$$

Algorithm 2

Step 1. Generate n random numbers from exponential distribution with parameter $\theta = 1$, $(x_i \sim \exp(1), i = 1, \ldots, n)$.

Step 2. Apply the following formula and deliver $\lambda = (\lambda_1, \ldots, \lambda_n)$ as a vector distributed uniformly on the $\Omega_{\mathbf{n}}$:

$$\lambda = (\lambda_1, \ldots, \lambda_n) = \left(\frac{x_1}{\sum_{i=1}^{n} x_i}, \ldots, \frac{x_n}{\sum_{i=1}^{n} x_i} \right).$$

David (1981) and Devroye (1986) discussed the properties of random numbers generated by first algorithm. Fang et al. (1997) investigated the properties of uniform random numbers generated on a convex polyhedron and its applications in experimental design. Onn and Weissman (2009) derived explicit expressions for the first moment of the largest spacing for random variables generated by second algorithm and used that to examine the validity of random number generators. Tian et al. (2009) constructed a one-to-one transformation between an arbitrary tetragon and the unit square. Uniform random numbers over a 3-dimensional simplex is input of their algorithm. They used Algorithm 1 to generate random numbers over a 3-dimensional simplex and applied their algorithm to the experimental design for a drug combination. Narayanan (1990) proposed an algorithm to generate random numbers from Dirichlet distribution, where in the one of the steps they generates uniform random numbers over a unit simplex by using first algorithm (proposed by Rubinstein, 1982). Hung et al. (2010) evaluated different methods in generating Dirichlet random vectors, in two of considered algorithms generating uniform random numbers over a unit simplex is required, but the algorithm used in this purpose was not discussed in their article.

To the best of our knowledge, there are two algorithms (Algorithms 1 and 2) in the literature for generating random numbers over a simplex. Since Inverse-Transformation method is the most well-known method in generating random numbers in simulation applications, in this article we aim to apply Inverse-Transformation method to generate random numbers over a simplex. This would be worthwhile when the measurements of some variables are in hand. We analytically derive the required conditional distribution, and then apply Inverse-Transformation method to generate random numbers over a simplex by having derived conditional distributions. We also evaluate the performance of the proposed method and two existing algorithms. Although theoretical results on Algorithms 1 and 2 have been well developed by Devroye (1986) and Onn and Weissman (2009), not much work has been done specifically on the computer generation time and goodness-of-fit criteria. The rest of the article is organized as follows. In the next section, we explain the proposed method. In Sec. 3, the proposed method is illustrated in two numerical examples. In Sec. 4, we evaluate the performance of the proposed method and two existing methods in terms of the following criteria: (i) computer generation times; (ii) goodness of fit. Finally, some concluding remarks are made in Sec. 5.

2. The Proposed Method

To generate a uniform random sample vectors of Ω_n we suppose:

$$\lambda_i \sim Uniform(0, 1), \quad i = 1, \ldots, n;$$

therefore:

$$f_{\lambda_i}(x) = 1, \quad 0 \le x_i \le 1, \ \ i = 1, \ldots, n$$
$$f_{\lambda_1,\ldots,\lambda_n}(x_i, \ldots, x_n) = 1, \quad 0 \le x_i \le 1, \ \ i = 1, \ldots, n$$

and each subset of a larger set with uniform distribution has uniform distribution. To generate uniform random numbers over Ω_n, we need to generate a random vector from the following distribution:

$$f_{\lambda_1,\ldots,\lambda_n}\left(x_1, \ldots, x_n \,\middle|\, \sum_{i=1}^{n} \lambda_i = 1\right).$$

To achieve this purpose, we require the conditional distribution of λ_1 given $\sum_{i=1}^{n} \lambda_i = 1$, the conditional distribution of λ_2 given $\sum_{i=1}^{n} \lambda_i = 1$ and λ_1, and the conditional distribution of λ_i given $\sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_1, \ldots, \lambda_{i-1}$.

By deriving these conditional distributions, the intended random numbers can be generated according to the Inverse-Transform method. In Inverse-Transform method, λ_1 is generated by inversing $F_{\lambda_1}(x_1 \mid \sum_{i=1}^{n} \lambda_i = 1)$ and λ_2 is generated from $F_{\lambda_2}(x_2 \mid \sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_1)$. $\lambda_i (i = 1, 2, \ldots, n-1)$ is generated from $F_{\lambda_i}(x_i \mid \sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_1, \ldots, \lambda_i)$ and finally λ_n is calculated as $1 - \sum_{i=1}^{n-1} \lambda_i$.

In the following, we first obtain the conditional distribution of λ_i's and then present a closed formula to generate λ_i's based on Inverse-Transform method.

2.1. *The Conditional Distributions of λ_i's*

First, we derive conditional probability density function of λ_1 given $\sum_{i=1}^{n}\lambda_i = 1$:

$$f_{\lambda_1}\left(X = x \,\middle|\, \sum_{i=1}^{n}\lambda_i = 1\right) = \frac{f_{\lambda_1,\sum_{i=1}^{n}\lambda_i}(x,1)}{f_{\sum_{i=1}^{n}\lambda_i}(1)} = \frac{f_{\sum_{i=1}^{n}\lambda_i}(1 \mid \lambda_1 = x) f_{\lambda_1}(x)}{f_{\sum_{i=1}^{n}\lambda_i}(1)} = \frac{f_{\sum_{i=2}^{n}\lambda_i}(1-x)}{f_{\sum_{i=1}^{n}\lambda_i}(1)}.$$

To find $f_{\sum_{i=1}^{n}\lambda_i}(1)$, we first obtain the cumulative density function (cdf) of $\sum_{i=1}^{n}\lambda_i$ as shown below:

$$\begin{aligned} F_{\sum_{i=1}^{n}\lambda_i}(1) &= \int_{\sum_{i=1}^{n}\lambda_i < X,\, 0\le X\le 1} f_{\lambda_1,\ldots,\lambda_n}(x_1,\ldots,x_n)\,d\lambda_1 d\lambda_2 \ldots d\lambda_n \\ &= \int_{\sum_{i=1}^{n}\lambda_i < X,\, 0\le X\le 1} 1\, d\lambda_1 d\lambda_2 \ldots d\lambda_n \\ &= \text{Volume of simplex with vertices } \ (\vec{0}, \vec{\lambda}_1, \vec{\lambda}_2, \ldots, \vec{\lambda}_n) \\ &\qquad \text{in } n\text{-dimensional space,} \end{aligned}$$

where $\vec{\lambda}_1 = [\lambda_{i,1} = 0, \ldots, \lambda_{i,i-1} = 0, \lambda_{i,i} = X, \lambda_{i,i+1} = 0, \ldots, \lambda_{i,n} = 0]$

However, the volume of an n-dimensional simplex in $\mathbb{R}^n$ with vertices $(v_0, v_1, \ldots, v_n)$ is: (Rudin, 1976)

$$= \frac{1}{n!}\left|\det(v_1 - v_0, v_2 - v_0, \ldots, v_n - v_0)\right|.$$

Then, volume of simplex with vertices $(\vec{0}, \vec{\lambda}_1, \vec{\lambda}_2, \ldots, \vec{\lambda}_n)$ is:

$$\frac{1}{n!}\left|\det([\vec{\lambda}_1, \vec{\lambda}_2, \ldots, \vec{\lambda}_n])\right| = \frac{X^n}{n!}$$

Therefore, the cumulative distribution function (cdf) of $\sum_{i=1}^{n}\lambda_i$ obtained by

$$F_{\sum_{i=1}^{n}\lambda_i}(X) = \frac{X^n}{n!}, \quad \text{only when } 0 \le X \le 1$$

and it yields

$$f_{\sum_{i=1}^{n}\lambda_i}(X) = \frac{\partial F_{\sum_{i=1}^{n}\lambda_i^{(x)}}}{\partial x} = \frac{x^{n-1}}{(n-1)!}, \quad 0 \le x \le 1,$$

where for $x = 1$,

$$f_{\sum_{i=1}^{n}\lambda_i}(1) = \frac{1}{(n-1)!}$$

and

$$f_{\lambda_1}\left(x \,\middle|\, \sum_{i=1}^{n}\lambda_i = 1\right) = \frac{\frac{(1-x)^{n-2}}{(n-2)!}}{\frac{1}{(n-1)!}} = (n-1)(1-x)^{n-2}, \quad 0 \le x \le 1$$

$$F_{\lambda_1}\left(x \,\middle|\, \sum_{i=1}^{n}\lambda_i = 1\right) = 1 - (1-x)^{n-1}, \quad 0 \le x \le 1.$$

Further, the conditional density functions can be shown to read as follows: ($k = 2, \ldots, n-1$)

$$f_{\lambda_k}\left(x \,\middle|\, \sum_{i=1}^{n} \lambda_i = 1, \lambda_1 = x_1, \lambda_2 = x_2, \ldots, \lambda_{k-1} = x_{k-1}\right)$$
$$= \frac{f_{\lambda_k \sum_{i=1}^{n} \lambda_i}(1 - x - \sum_{i=1}^{k-1} \lambda_i, 1)}{f_{\sum_{i=1}^{n} \lambda_i}(1 - \sum_{i=1}^{k-1} \lambda_i)} = \frac{\frac{(1-x-\sum_{i=1}^{k-1} \lambda_i)^{n-k-1}}{(n-k-1)!}}{\frac{(1-\sum_{i=1}^{k-1} \lambda_i)^{n-k}}{(n-k)!}}$$
$$= \frac{n-k}{(1 - \sum_{i=1}^{k-1} \lambda_i)^{n-k}}\left(1 - x - \sum_{i=1}^{k-1} \lambda_i\right)^{n-k-1};$$

we thus have

$$F_{\lambda_k}\left(x \,\middle|\, \sum_{i=1}^{n} \lambda_i, = 1, \lambda_1 = x_1, \lambda_2 = x_2, \ldots, \lambda_{k-1} = x_{k-1}\right)$$
$$= \int_0^x f_{\lambda_k}\left(t \,\middle|\, \sum_{i=1}^{n} \lambda_i, , = 1, \lambda_1 = x_1, \lambda_2 = x_2, \ldots, \lambda_{k-1} = x_{k-1}\right) dt$$
$$= 1 - \left(1 - \frac{x}{1 - \sum_{i=1}^{k-1} \lambda_i}\right)^{n-k}.$$

2.2. *Inverse-Transform Method*

Let be a random variable with cdf $F_x(x)$. Since $F_x(x)$ is a non decreasing function, the inverse function of $F_x(x)$ may be defined as $F^{-1}(y) = \{x : F(x) = y\}$, $0 \le y \le 1$. It is easy to show that if $U \sim Uniform(0, 1)$, then has $X = F^{-1}(U)$ has cdf $F_x(x)$. (see Hogg and Craig, 1970, for more details).

Namely, since F is invertible and $P(U < u) = u$, we have:

$$P(X \le x) = P(F^{-1}(U) \le x) = P(U \le F(x)) = F(x).$$

Therefore, to generate a random variable X with df $F_x(x)$, draw $U \sim Uniform(0, 1)$ and set $X = F^{-1}(U)$.

The steps of the Inverse-Transform method are:

Step 1. Generate uniform random number from $Uniform(0, 1)$;
Step 2. Return $X = F^{-1}(U)$.

2.3. *Uniform Variables Generation Over Ω_n*

Thus, by inversing the $F_{\lambda_1}(x \mid \sum_{i=1}^{n} \lambda_i = 1) = (1 - \lambda_1)^{n-1}$, random values for λ_1 can be found by using the following transformation:

$$\lambda_1 = 1 - (1 - U_1)^{1/n-1}.$$

Since $(1 - U_1)$ and U_1 have same distribution ($U_1 \sim Uniform(0, 1)$ and $(1 - U_1) \sim Uniform(0, 1)$), λ_1 can be generated in below form:

$$\lambda_1 = 1 - U_1^{1/n-1}.$$

For $k = 2, \ldots, n-1$ the following transformation based on obtained conditional cdf is:

$$\lambda_k = \left(1 - \sum_{i=1}^{k-1} \lambda_i\right)\left(1 - U_k^{\frac{1}{n-k}}\right) = U_1^{1/n-1} \ldots U_{k-1}^{1/n-(k-1)}\left(1 - U_k^{1/n-k}\right)$$
$$= \left[\prod_{j=1}^{k-1} U_j^{\frac{1}{n-j}}\right]\left(1 - U_k^{\frac{1}{n-k}}\right).$$

Finally, λ_n is computed as

$$\lambda_n = 1 - \sum_{i=1}^{n-1} \lambda_i.$$

2.4. *Uniform Variate Generation Over a Simplex*

In the previous section, we generate uniform random variates over a unit standard simplex. In order to generate random numbers uniformly distributed over an n-dimensional simplex $\mathbf{S_n}$ defined by arbitrary vertices, say $Z_1, \ldots, Z_n$, we simply generate $(\lambda_1, \ldots, \lambda_n)$ uniformly distributed on $\Omega_\mathbf{n} = \{\lambda : \lambda \geq 0, i = 1, \ldots, n \text{ and } \sum_{i=1}^n \lambda_i = 1\}$ using the formulae extracted in Sec. 2.3 and apply the following linear transformation to get numbers uniformly distributed over $\mathbf{S_n}$:

$$\gamma = \mathbf{C}\lambda,$$

where $\mathbf{C}$ is the matrix whose columns are $Z_1, \ldots, Z_n$.

3. Numerical Examples

In this section, we provide two numerical examples to illustrate the developed method.

Example 3.1. In this example, we generate 500 uniform random data over a simplex with vertices (1, 2, 3), (3, 1, 2), (1, 4, 10). First, we generate 500 random vectors of $\Omega_3 = \{\lambda : \lambda_i \geq 0, i = 1, 2, 3 \text{ and } \sum_{i=1}^3 \lambda_i = 1\}$ by using the following formulae:

$$\lambda_1 = 1 - U_1^{1/3-1}$$
$$\lambda_2 = 1 - U_1^{1/3-1}(1 - U_2^{1/3-2})$$
$$\lambda_3 = 1 - U_1^{1/3-1}(U_2^{1/3-2}).$$

(U_1 and U_2 are uniform random numbers generated from $Uniform(0, 1)$). Finally, each random vector is calculated by:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 1 & 4 \\ 3 & 2 & 10 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

Figure 1 shows these 500 random data in this example.

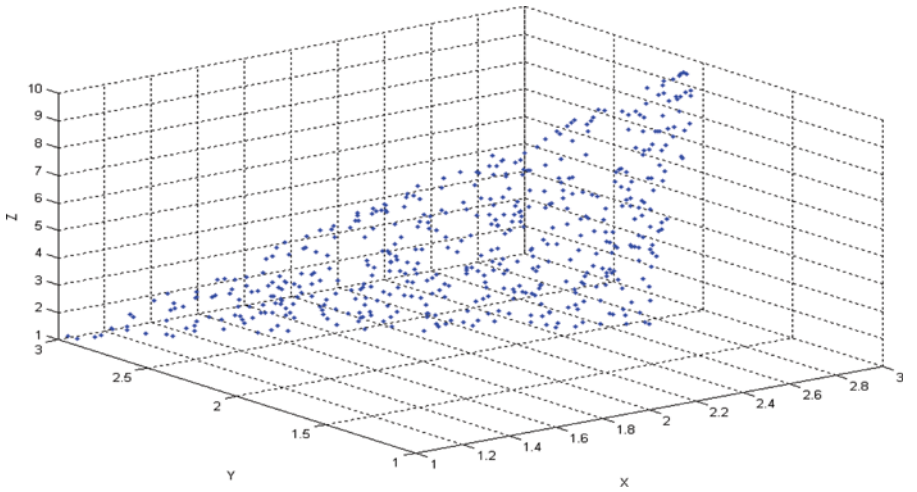

Figure 1. Randomly generated data in Example 3.1.

Example 3.2. In this example, we generate uniform random data in a simplex with vertices (1, 1, 0), (2, 3, 0), (3, 2, 0), (2, 2, 3). In this example, λ_is ($i = 1, 2, 3$ and 4) are generated from:

$$\Omega_4 = \left\{ \lambda : \lambda_i \geq 0, \quad i = 1, \ldots, 4 \text{ and } \sum_{i=1}^{4} \lambda_i = 1 \right\}$$

and random data on the simplex with vertices (1 1 0), (2 3 0), (3 2 0), (2 2 3) are computed as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}.$$

Figure 2 shows the 10,000 generated data in this example.

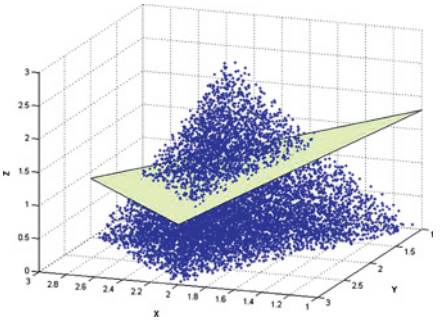

Figure 2. Randomly generated data in Example 3.2, cutting face is $3x + 3y + 10z = 24$.

4. Performance Evaluation

In this section, we evaluate performance of the proposed method with existing algorithms in terms of computer generation time and goodness of fit criteria. Table 1 shows computer generation times for different methods. It can be seen from Table 1 that, in all simulation runs, Algorithm 2 has the smallest computer time (in second) while computer time required by the proposed algorithm is consistently less than that of Algorithm 1 in all cases. (All the algorithms are implemented in MATLAB 7.1 on a PC equipped with INTEL 2 Core, 1.86-GHz CPU and 512 MB RAM memory.)

Another way of evaluating a random generator algorithm is to perform a goodness-of-fit test based on the generated random numbers. We perform Kolmogorov-Smirnov (K-S) test developed by Anderson (1966).

To perform K-S test, we generate 1,000 vectors $Y_i = [y_{1i}, y_{2i}, y_{3i}]$, $i = 1, 2, \ldots, 1{,}000$, by using each algorithm over a 3-dimensional unit simplex. We need to divide the area into m ($m = 1{,}000$) equal areas called block. The statistically equivalent blocks are defined as:

$$B_j = \left\{ \begin{array}{l} \dfrac{(j-1)}{500} y_1 \le y_2 < \dfrac{j}{500} y_1,\ 0 \le y_3 \le 1,\ \displaystyle\sum_{i=1}^{3} y_i,\ j = 1, \ldots, 500 \\ \dfrac{(j-500-1)}{500} y_2 \le y_1 \dfrac{j-500}{500} y_2,\ 0 \le y_3 \le 1,\ \displaystyle\sum_{i=1}^{3} y_i = 1,\ j = 501, \ldots, 1000 \end{array} \right\}.$$

By this definition, the area of each block is $\frac{\sqrt{3}}{\sqrt{2}} \times \frac{\sqrt{2}}{1000} = 0.000866$. Now, we expect to have equal numbers of generated data in each block. d_j denotes number of generated data in block j, and $D_j = \sum_{i=1}^{j} d_i$. The Kolmogorov-Smirnov statistics ($\widetilde{D}_n$) is defined as

$$\widetilde{D}_n = \max\left\{0,\ \max\left(\frac{j}{n} - D_j\right),\ \max\left(D_j - \frac{j-1}{n}\right),\ \text{for all } j = 1, \ldots, n\right\}.$$

Under the null hypothesis, the 95th percentile of N resulting $\sqrt{n}\widetilde{D}_n$ is very close to 0.95. We refer interested readers to Hung et al. (2010) for more detail on this test where they evaluate different random generator algorithms. We set $N = 200$, means we replicate generating 1,000 data by each method 200 times. Then we calculate 95th percentile of 200 resulting $\sqrt{n}\widetilde{D}n$ for each method. The method that presents a probability index close to 0.95 is better. The probability index of goodness of fit test of the proposed method is 0.945 when it is 0.941 and 0.942 for Algorithms 1 and 2,

Table 1
The computer generation times (in seconds) for different methods to generate 10^5 vectors over a unit simplex

Dimension	3	5	10	20	50	500	1000	10000
Proposed method	0.598	0.643	0.732	0.883	1.225	11.737	22.896	248.824
Algorithm 1	0.483	0.530	0.624	0.733	1.060	6.630	13.385	156.141
Algorithm 2	0.639	0.733	0.748	0.920	1.404	12.027	24.305	253.735

respectively. This result indicates that the proposed method exhibits more accuracy in terms of goodness of fit.

5. Conclusion

In this article, a method for uniform random data generation over a simplex was presented based on derived conditional distributions. Comparison results with two existing algorithms shows the proposed method needs less computational time than one of the existing algorithms. However, by considering goodness-of-fit test for generated vectors by each method, the proposed method provides more accuracy. Besides, when one needs to generate k elements of λ_i's ($k < n$) the proposed method needs only k random numbers while in the existing methods we need n random numbers. The proposed method can be applied to generate uniform random numbers (λ's) over a simplex when the measurements of some other λ's are available.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which substantially improved the article.

References

Anderson, T. W. (1966). Some nonparametric procedures based on statistically equivalent blocks. In: Krishnaiah, P. R., ed. *Proceedings of International Symposium on Multivariate Analysis*. New York: Academic Press Inc.

David, H. A. (1981). *Order Statistics*. New York: Wiley.

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

Fang, K. T., Tian, G. L., Xie, M. Y. (1997). Uniform distribution on convex polyhedron and its applications. Mathematics Department, Hong Kong Baptist University, *Technical Report No. 149*, Hong Kong.

Hogg, R. V., Craig, A. T. (1970). *Introduction to Mathematical Statistics*. London: MacMillan.

Hung, Y. C., Balakrishnan, N., Cheng, C. W. (2010). Evaluation of algorithms for generating Dirichlet random vectors. *Journal of Statistical Computation and Simulation*, [published online June 2010].

Narayanan, A. (1990). Computer generation of Dirichlet random vectors. *Journal of Statistical Computation and Simulation* 36:19–30.

Onn, S., Weissman, I. (2009). Generating uniform random vectors over a simplex with implications to the volume of a certain polytope and to multivariate extremes. *Annals of Operations Research*, [published online 2009].

Rubinstein, R. Y. (1982). Generating random vectors uniformly distributed inside and on the surface of different regions. *European Journal of Operational Research* 10:205–209.

Rudin, W. (1976). *Principles of Mathematical Analysis*. 3rd ed. Ch. 10. New York: McGraw-Hill.

Tian, G. L., Fang, H. B., Tan, M., Qin, H., Tang, M. L. (2009). Uniform distributions in a class of convex polyhedrons with applications to drug combination studies. *Journal of Multivariate Analysis* 100(8):1854–1865.

Vetschera, R., Chen, Y., Hipel, K. W., Kilgour, D. M. (2010). Robustness and information levels in case-based multiple criteria sorting. *European Journal of Operational Research* 202(3):841–852.