A New Method to Generate Uniform Random Variates on the Unit Spherical Shell in $\mathbb{R}^d$

Bárbara González-Arévalo[a] and Abel Rodríguez[b]

[a]Northern Illinois University, 1425 Lincoln Hwy, DeKalb, IL 60115; [b]University of Washington, 1410 NE Campus Parkway Seattle, WA 98195

ARTICLE HISTORY
Compiled December 20, 2024

ABSTRACT
The generation of uniform random variates on the unit spherical shell is a key step in the simulation of many useful densities such as the multivariate normal and the multivariate Student's t. We present a novel algorithm to generate samples that are uniformly distributed on the unit spherical shell. Our method relies on recursively writing the joint density of the vector of random variables as the product of the marginal distribution of one component and the corresponding conditional distribution for the rest of the components, which after rescaling turns out to correspond to the density of another uniformly distributed random variable on a unit spherical shell of lower dimension. The results presented here can be especially useful for extending existing algorithms, such as the Box-Muller method.

KEYWORDS
Probability, Simulation, Random Variate Generation; Isotropic Density; Accept-Reject Method; Ratio-of-Uniforms Method.

1. Introduction

The generation of uniform random variates on the unit spherical shell is a critical step in simulating isotropic and ellipsoidally symmetric distributions.

An isotropic density $\psi(x)$ is defined as one that is invariant to rotations. That is, a density ψ is isotropic if $\psi(x) = \phi(||x||)$ for some function ϕ. Through a change of variables to polar coordinates, isotropic densities can be written as the product of two other random variables, one of them uniformly distributed on the unit spherical shell (see, for example, Mardia & Jupp (2000)). Hence, generating random variables in the unit spherical shell is a key step in the simulation of random variates that follow isotropic (and, more generally, ellipsoidally symmetric) distributions such as the multivariate normal and the multivariate Student's t.

These distributions, essential in fields such as physics (e.g. seen in simulations such as molecular dynamics where uniformity of direction is crucial in Frenkel & Smit (2002)), computer graphics (e.g. for rendering realistic visuals involving lighting in Pharr, Jakob, & Humphreys (2016)), and machine learning (e.g. used in algorithms requiring isotropic priors or kernel methods in Bishop (2006)), require efficient sampling techniques to accurately model phenomena involving directional data.

CONTACT Bárbara González-Arévalo. Email:barbara.p.gonzalez@gmail.com

One of the most influential methods for generating these random variables is based on scaling vectors derived from multivariate normal distributions. This approach was popularized by Marsaglia (1972) with his paper "Choosing a Point from the Surface of a Sphere," where he discussed generating a vector of independent standard normal variables and normalizing them to project onto the unit sphere.

Muller (1959) in "A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres," provided fundamental insights into generating points in higher-dimensional spheres, laying the groundwork for subsequent algorithmic developments.

An alternative approach involves the use of spherical coordinates, as detailed in methods discussed by Devroye (1986) in his book "Non-Uniform Random Variate Generation." This technique involves generating random angles uniformly, converting these into Cartesian coordinates. In higher dimensions, this method becomes complex due to the multidimensional nature of angle generation.

We define these methods in Section 2.

The comparative effectiveness of these methods generally balances between computational speed and accuracy. The scaling normal vectors method is favored for its robustness and simplicity, particularly in high-dimensional applications. In contrast, the spherical coordinate approach, while conceptually straightforward, can become cumbersome with increased dimensions, impacting computational efficiency.

In Section 3 of this paper we discuss an alternative approach where we generate directly from the distribution. Our method relies on recursively writing the joint density of the vector of random variables as the product of the marginal distribution of one component and the corresponding conditional distribution for the rest of the components, which after rescaling turns out to correspond to the density of another uniformly distributed random variable on a unit spherical shell of lower dimension.

In Section 4, we discuss the possible methods to generate directly from the desired distribution. These methods include an inverse transform method where the inverse distribution function is defined, a method involving the transformation of Beta and Gamma random variables, an accept reject method involving the Ziggurat Method, and a ratio-of-uniforms method which outperforms all of the other methods.

In Section 5, we discuss the implementation of each algorithm, and in Section 6, we examine the numerical results of the comparison of these algorithms.

2. Algorithms Currently Being Used

2.1. Scaling of Normal Vectors

The following results depend on the fact that vectors containing standard Gaussian random variables have an isotropic behavior. If we rescale a normal vector of size d to have a norm of 1, then the new vector will also be isotropically distributed, and it will be on the sphere of dimension d. Thus, the algorithm is based on the below steps:

(1) Generate a vector of size d with normal random variables
(2) Find the magnitude of the vector
(3) Divide by the magnitude so the new vector has a magnitude of one

Note that this method depends on the ability to generate random variables from the Gaussian distribution Martinez (2008).

2.2. Spherical Coordinates

Generating on the d-dimensional unit spherical shell is also equivalent to generating a vector of the form:

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_{d-1} \\ U_d \end{pmatrix} = \begin{pmatrix} \sin\theta_1 \sin\theta_2 \cdots \sin\theta_{d-2} \sin\theta_{d-1} \\ \sin\theta_1 \sin\theta_2 \cdots \sin\theta_{d-2} \cos\theta_{d-1} \\ \sin\theta_1 \sin\theta_2 \cdots \cos\theta_{d-2} \\ \cdots \\ \sin\theta_1 \cos\theta_2 \\ \cos\theta_1 \end{pmatrix} \tag{1}$$

where $\Theta = (\theta_1, \ldots, \theta_{d-1})$ is a random vector with marginal densities $f(\theta_i)$ that are proportional to $\sin^{d-i-1}\theta_i$ for $1 \leq i \leq d-2$, and θ_{d-1} is generated uniformly from the interval $(0, 2\pi)$. See Boisbunon (2011) for a more detailed discussion of this method.

3. Theoretical Background and Basic Algorithm

Our algorithm is based on a recursive decomposition of the joint density of the vector of random variables as the product of the marginal distribution of one component and the corresponding conditional distribution for the rest of the components, which after rescaling turns out to correspond to the density of another uniformly distributed random variable on a unit spherical shell of lower dimension. More specifically, let $B_d(r) = \left\{x \in \mathbb{R}^d \mid ||x|| = r\right\}$ be the spherical shell of radius r in $\mathbb{R}^d$ and $\Xi = (\xi_1, \ldots, \xi_d) \in \mathbb{R}^d$ with $\Xi \sim \mathsf{Unif}\left[B_d(1)\right]$. Our algorithm relies on decomposing the uniform density of Ξ as

$$f(\xi_1, \ldots, \xi_{i-1}, \xi_i, \xi_{i+1} \ldots, \xi_d) = f(\xi_1, \ldots, \xi_{i-1}, \xi_{i+1} \ldots, \xi_d \mid \xi_i) f(\xi_i),$$

and exploits the fact that, for any i, the first term of the decomposition reduces (after an appropriate rescaling) to a uniform on $B_{d-1}(1)$. Applying this idea recursively leads to an algorithm for simulating from the uniform distribution on the unit spherical shell that only requires us to generate from the marginal distribution of a single component of Ξ for every value of d.

The following results for Ξ can be easily established and underpin the algorithm that we will present in Proposition 4.

Proposition 1. For all fixed $d \geq 2$ the marginal density of ξ_i is

$$h(\xi_i \mid d) = \begin{cases} \frac{\omega_{d-1}}{\omega_d}(1-\xi_i^2)^{\frac{d-3}{2}} & -1 \leq \xi_i \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where ω_d is the area of the unit spherical shell on $\mathbb{R}^d$.

Proof. Straightforward and omitted. To find the marginal density of ξ_i, integrate the joint density over all other coordinates. □

The case $d = 1$ (which is not included in Proposition 1) is trivial, since in that case the distribution of ξ_1 reduces to two equally weighted points of mass, one located at

$\xi_1 = -1$ and another at $\xi_1 = 1$. Note also that $h(\xi_i \mid d)$ corresponds to the uniform density only for $d = 3$.

Proposition 2. Let $\xi_{-i} = (\xi_1, \ldots, \xi_{i-1}, \xi_{i+1}, \ldots, \xi_d)$. The conditional density $f(\xi_{-i} \mid \xi_i)$ is the uniform on $B_{d-1}\left(\sqrt{1-\xi_i^2}\right)$.

Proof. First note that since the support of ξ is $B_d(1)$, the support of $\xi_{-i} \mid \xi_i$ is $B_{d-1}\left(\sqrt{1-\xi_i^2}\right)$. On the other hand, since $B_{d-1}\left(\sqrt{1-\xi_i^2}\right) \subset B_d(1)$, then $\xi_{-i} \mid \xi_i$ is uniformly distributed on this space. □

Proposition 3. If $\Xi \sim \mathsf{Unif}[B_d(r)]$ and $\Psi = t \cdot \Xi$ for any $t \in \mathbb{R}$, then $\Psi \sim \mathsf{Unif}[B_d(tr)]$.

Proof. Straightforward from a change of variables. □

Proposition 2 shows that conditioning on any component of the random vector Ξ keeps the remaining components on a spherical shell (albeit one that lives in a lower-dimensional space and has a smaller radius). On the other hand, Proposition 3 implies that to generate random variates from $f(\xi_{-i} \mid \xi_i)$ we only need to be able to simulate from a uniform distribution on the unit spherical shell in dimension $d - 1$. Putting these two results together suggests an iterative procedure that generates random variates from the unit spherical shell by recursively sampling from density (2) and appropriately rescaling the resulting random variables. Such procedure is formalized in the following Proposition 4.

Proposition 4. The algorithm defined by the steps

(1) Initialize $r_0 = 1$
(2) For all $i = 1, \ldots, d-2$:
    (a) Generate $\xi_i^* \sim h(\xi_i^* \mid d - i + 1)$, where h was defined in Equation (2).
    (b) Calculate $\xi_i = r_{i-1} \cdot \xi_i^*$
    (c) Set $r_i = \sqrt{r_{i-1}^2 - \xi_i^2}$
(3) Set $\xi_{d-1} = r_{d-2}\xi_{d-1}^*$ and $\xi_d = r_{d-2}\xi_d^*$, where $(\xi_{d-1}^*, \xi_d^*) \sim \mathsf{Unif}[B_2(1)]$

generates uniform random variates on $B_d(1)$.

Proof. Straightforward from Propositions 1, 2, and 3. □

4. Methods Involving the Basic Algorithm from Section 3

In order to completely specify the previous algorithm, it is necessary to establish a method to generate the random variates mentioned in step 2a. In what follows, we review five such methods: inversion, transformation, naive acceptance-rejection, ziggurat acceptance-rejection, and ratio-of-uniforms. In general, each of these methods has advantages and disadvantages that we will point out as we describe the methods.

4.1. Random Variate Generation via Inversion

The inversion method for generating random variables offers the advantage of simplicity and efficiency when the inverse cumulative distribution function (CDF) is easily obtainable. It provides a direct and straightforward way to generate random variables

from a given distribution by applying the inverse CDF to uniform random variables. However, its main disadvantage is that it requires the CDF to have a closed-form inverse, which is not always available. In cases where the inverse CDF is difficult to compute or cannot be expressed analytically, the method becomes computationally expensive or even impractical. Thus, while it is an attractive option when applicable, its utility is limited to distributions with easily invertible CDFs.

Therefore, this method will work to generate the random variables in step 2a for $d < 4$, since when $d \geq 4$ the distributions are not analytically invertible.

To use this method, we will need the distribution function corresponding to the density in (2).

Proposition 5. Let X be a random variable with density (2). If we let $m = \left\lfloor \frac{d-2}{2} \right\rfloor$, the corresponding distribution function for all $d \geq 2$ is

$$F(x) = \begin{cases} 0 & x < -1 \\ \frac{1}{2} + \frac{\omega_{d-1}}{\omega_d} \left[(d-3-2m)a(m)x + \sum_{i=0}^{m} a(i)x \left(1-x^2\right)^{\frac{d-2i-3}{2}} \right] & -1 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$$

for odd d and

$$F(x) = \begin{cases} 0 & x < -1 \\ \frac{1}{2} + \frac{\omega_{d-1}}{\omega_d} \left[(d-3-2m)a(m)\sin^{-1} x + \sum_{i=0}^{m} a(i)x \left(1-x^2\right)^{\frac{d-2i-3}{2}} \right] & -1 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$$

for even d. In the expressions above, $\lfloor x \rfloor$ represents the largest integer less than or equal to x and

$$a(i) = \frac{\prod_{j=0}^{i-1}(d-3-2j)}{\prod_{j=0}^{i}(d-2-2j)}.$$

Proof. By definition, for $-1 \leq x \leq 1$,

$$\begin{aligned} F(x) = \Pr(X < x) &= \int_{-\infty}^{x} f(\xi)d\xi \\ &= \int_{-1}^{x} \frac{\omega_{d-1}}{\omega_d}(1-\xi^2)^{\frac{d-3}{2}}d\xi \\ &= \frac{1}{2} + \int_{0}^{x} \frac{\omega_{d-1}}{\omega_d}(1-\xi^2)^{\frac{d-3}{2}}d\xi, \end{aligned}$$

since the density is symmetric around zero. Using the change of variables $\xi = \sin t$ and integrating by parts, it is easy to see that

$$\int_0^x (1-\xi^2)^{\frac{d-3}{2}}d\xi = \frac{x(1-x^2)^{\frac{d-3}{2}}}{d-2} + \frac{d-3}{d-2}\int_0^x (1-\xi^2)^{\frac{d-5}{2}}d\xi.$$

Using this equation recursively together with the terminal relationships

$$\begin{aligned}\int_0^x (1-\xi^2)^0 d\xi &= x \\ \int_0^x (1-\xi^2)^{-\frac{1}{2}} d\xi &= \sin^{-1} x\end{aligned}$$

yields the desired result. □

For $d = 2$ and $d = 3$ the distributions are

$$F(x \mid d=2) = \frac{1}{2} + \frac{\sin^{-1} x}{\pi}, \text{ and} \tag{3}$$

$$F(x \mid d=3) = \frac{1}{2} + 2x, \tag{4}$$

and their inverses are

$$F^{-1}(u \mid d=2) = \cos(\pi u), \text{ and} \tag{5}$$

$$F^{-1}(u \mid d=3) = 2u - 1. \tag{6}$$

4.2. A Transformation Algorithm

The transformation method for generating random variables is flexible and can be applied to a wide range of distributions if an appropriate transformation function is known. When a good transformation exists, this method can be very efficient, as it allows random variables from a simpler, easily generated distribution to be transformed into the desired distribution. However, its main disadvantage is that it requires a known and computable transformation function, which is not always available for complex or non-standard distributions. Additionally, deriving or identifying the transformation for certain distributions can be challenging, making the method less practical in cases where no simple transformation exists.

In our case, the density shown in (2) belongs to the Pearson II family (Johnson, Kotz, & Balakrishnan , 1994) and therefore, it is related to the beta density in the following way:

Proposition 6. Let X and Z be two random variables such that $X = 2Z - 1$. Then X has density (2) if and only if $Z \sim \mathsf{Beta}\left(\frac{d-1}{2}, \frac{d-1}{2}\right)$, i.e. the density of Z is $f(z \mid d) = \frac{\Gamma(d-1)}{\Gamma\left(\frac{d-1}{2}\right)^2} z^{\frac{d-3}{2}} (1-z)^{\frac{d-3}{2}}$.

Proof. Straightforward from a change of variables. □

Thus, any algorithm for the generation of beta random variables can be used to produce random variables distributed according to the density in (2). For example, we could exploit the well-known relationship between the Beta and the Gamma densities:

Proposition 7. If $X, Y \sim \mathsf{Gamma}\left(\frac{d-1}{2}, a\right)$ with $d > 1$ and $a > 0$, then the density of $Z = \frac{X-Y}{X+Y}$ reduces to that in Equation (2).

Proof. Straightforward using Proposition 6 and the well known fact that $\frac{X}{X+Y} \sim \mathsf{Beta}\left(\frac{d-1}{2}, \frac{d-1}{2}\right)$. □

Note that using this property together with Proposition 4 is equivalent to the method described by Hicks & Wheeling (1959). Of particular interest for improving the performance of these algorithms are algorithms for generating Beta random variates that take advantage of the particular features of the distribution in (2), i.e. its symmetry and the form of its parameter (which is an integer multiple of 1/2).

4.3. The First Try At An Acceptance-Rejection Method

The acceptance-rejection method is simple to implement and highly versatile, as it can be applied to any target distribution, even when its probability density function is complex or unknown. By sampling from a proposal distribution and accepting or rejecting based on the target distribution, it offers a general approach to generating random variables. However, its main disadvantage is inefficiency, especially when the proposal distribution poorly matches the target distribution. This leads to high rejection rates, which can significantly increase the number of samples needed to obtain a valid random variable. As a result, it can become computationally expensive, particularly when the acceptance rate is low or the proposal distribution is not well chosen.

In order to design one of those algorithms for the density in (2), notice that since the density is symmetric around zero, it is enough to generate X using

$$f^*(x \mid d) = \begin{cases} 2\frac{\omega_{d-1}}{\omega_d}(1-x^2)^{\frac{d-3}{2}} & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and then assign its sign randomly with uniform probability. The density

$$g(x \mid s, c) = \frac{(1+s^c)^{\frac{1}{c}}}{s}\left[1+\left(\frac{x}{s}\right)^c\right]^{-\left(1+\frac{1}{c}\right)},\ 0 \le x \le 1,\ s > 0 \tag{8}$$

which belongs to the Burr III family (Burr, 1942; Burr & Cislak, 1968; Burr, 1973) was initially proposed as the blanketing function. Note that this choice is similar to that of Cheng (1978).

While this method does generate from the density f^*, the rejection probability at the tail is very large. Thus, it is suggested to instead use the Ziggurat Algorithm to generate from f^*.

4.4. The Ziggurat Algorithm Approach to An Acceptance-Rejection Method

The Ziggurat Algorithm, first proposed by (Marsaglia & Tsang, 2000), provides a quick method to generate from a decreasing probability density (in our case, f^*). The algorithm is based on the idea of filling our probability density with rectangular layers of equal area, and then generating from these rectangles. Most of the work involved in this method is completed when we set up the Ziggurat with auxiliary tables, and about 99% of the time, a point is generated from the desired density after just two table fetches and comparison of magnitude.

The Ziggurat Algorithm is extremely efficient because the area under the probability density divided by the area contained within the rectangular layers is very close to 1, which means that the rejection probability is extremely low.

Note that each d has a different setup of the ziggurat because each d has a distinct probability density. Because of this, there is a different f^* depending on the dimension we are looking at. A downside to using the Ziggurat Algorithm to generate from f^* at dimension d is that you would need to have auxiliary tables for each dimension less than d (since this is a recursive algorithm). There is a way to transform the values from a given dimension into any other dimension, but this method involves taking integrals and is very costly.

4.5. A Ratio-of-Uniforms Method

The ratio-of-uniforms method (Kinderman & Monahan, 1977) is an alternative acceptance-rejection method to generate from density (2). In this case, a pair (U, V) must be generated uniformly in the region

$$\Omega = \left\{ (u, v) \,\middle|\, v^2 \leq u^2 \left(1 - u^{2/r}\right),\ r = \frac{d-3}{2} \right\}$$

and then the quotient $Z = \frac{V}{U}$ follows the density in (2). There are two methods to generate from this region Ω.

4.5.1. Method 1: Rejection Sampling in Two Dimensions

The pair (U, V) can be obtained by rejection sampling from the rectangle

$$M = \left\{ (u, v) \,\middle|\, 0 \leq u \leq 1,\ |v| \leq \sqrt{\frac{r^r}{(1+r)^{1+r}}},\ r = \frac{d-3}{2} \right\}$$

which completely encloses Ω. Figure 1 shows the region Ω and the rectangle that we perform rejection sampling on.

The behavior of the rejection constant, which is the expected value of the number of iterations needed, for $4 \leq d \leq 100$ can be seen in Figure 2. The rejection constant is the expected value of a geometric random variable, $\frac{1}{p}$, where p is the ratio of the area of the acceptance region Ω over the area of the rectangle that encloses Ω. For $d = 4$, the constant is about 1.57, but decreases very fast and stabilizes around 1.37. Despite the fact that the rejection constants are quite high in this method, the evaluation of the rejection condition requires few operations.

Due to the shape of Ω, the squeeze function

$$f^*(u, v \mid d) = \begin{cases} |v| \leq \alpha_0 u & u \leq \left(\frac{r}{1+r}\right)^{r/2} \\ |v| \leq \alpha_1 (1-u) & u > \left(\frac{r}{1+r}\right)^{r/2} \end{cases} \tag{9}$$

with $\alpha_0 = \sqrt{1+r}$ and $\alpha_1 = \frac{r^{r/2}}{\sqrt{1+r}\left[(1+r)^{r/2} - r^{r/2}\right]}$ can provide a quick acceptance step. In Section 6, we discuss whether this squeeze function made the algorithm more efficient.

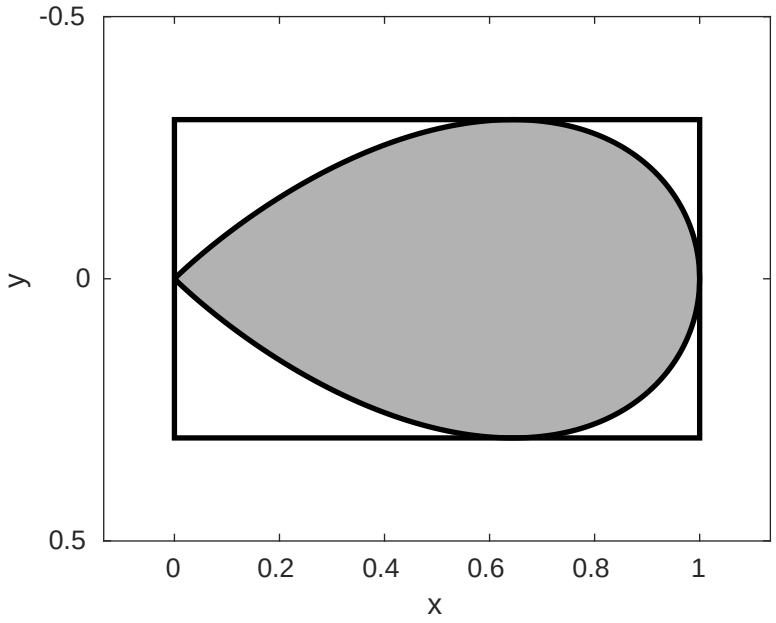

Figure 1. Example of a plot of the region Ω and the blanketing rectangle for $d = 10$.

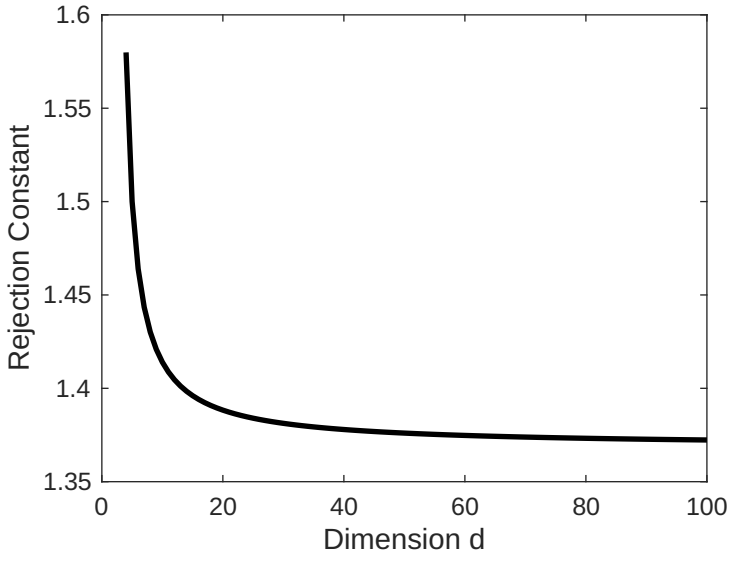

Figure 2. Plots of the rejection constant for the ratio-of-uniforms method up to $d = 100$.

4.5.2. Method 2: Rejection Sampling in One Dimension

Rather than doing an accept-reject on a point from the rectangle, we can do an accept-reject on the marginal distribution of U and then generate V uniformly given $U = u$.

Proposition 8. Let U be generated from the density $g(u) = \frac{2u\sqrt{1-u^{2/r}}}{|\Omega|}$. Then, $V|U = u \sim \mathsf{Unif}[-u\sqrt{1-u^{2/r}}, u\sqrt{1-u^{2/r}}]$.

Proof. In the region Ω, $v^2 \leq u^2(1-u^{2/r})$. After applying some simple algebra, we have the inequality $-u\sqrt{1-u^{2/r}} \leq v \leq u\sqrt{1-u^{2/r}}$. We know that:

$$g(v|u) = \frac{g(u,v)}{g(u)}$$
$$g(u,v) = \frac{1}{|\Omega|}$$

where $g(u) = \int_\Omega g(u,v)dv$ and $|\Omega|$ equals the area of Ω. Therefore, for $0 \leq u \leq 1$,

$$\begin{aligned} g(u) &= \int_{-u\sqrt{1-u^{2/r}}}^{u\sqrt{1-u^{2/r}}} \frac{1}{|\Omega|} dv \\ &= \frac{1}{|\Omega|} \int_{-u\sqrt{1-u^{2/r}}}^{u\sqrt{1-u^{2/r}}} dv \\ &= \frac{2u\sqrt{1-u^{2/r}}}{|\Omega|} \end{aligned}$$

So the conditional distribution of V given $U = u$ for $(u,v) \in \Omega$, equals:

$$g(v|u) = \frac{1}{2u\sqrt{1-u^{2/r}}} \sim \mathsf{Unif}[-u\sqrt{1-u^{2/r}}, u\sqrt{1-u^{2/r}}]$$

□

5. Implementation

Each method to generate variates on the d-sphere was implemented in MATLAB R2016B. All the code was run on an AMD A8-7410 processor with 2.20 GHz speed.

Note that we chose to implement the acceptance rejection method to generate from f^* using auxiliary tables for each dimension. This is because the transformation method was inefficient, and memory is cheap.

Also, note that the Ziggurat Algorithm was implemented in a bare-bones MATLAB program to create normal random variables for the scaling of normal vectors method and the Acceptance-Rejection method detailed in section 4.4. While MATLAB has its own built-in function to generate normal random variates, this function has been highly optimized over many years. Therefore, we chose to implement all of the algorithms ourselves for a fair comparison. The MATLAB code files for each method are

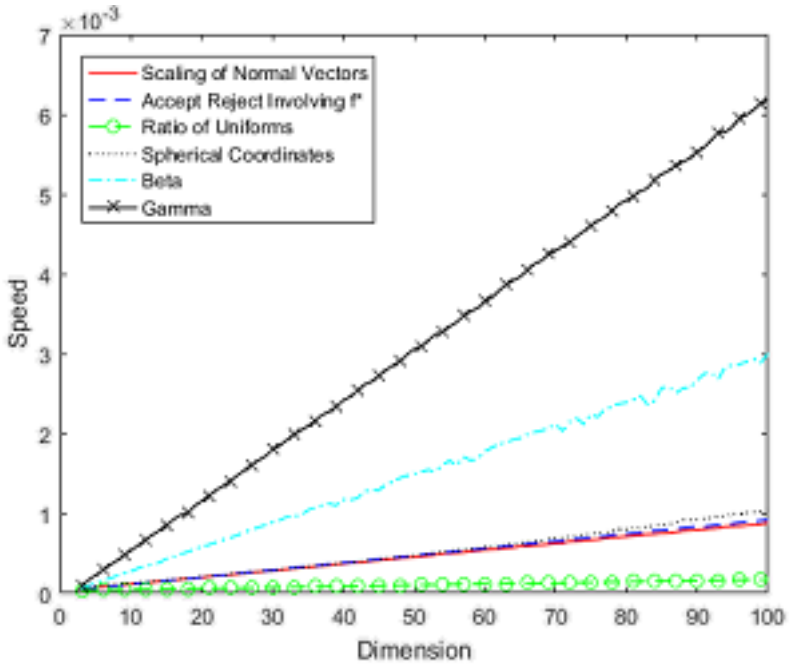

Figure 3. Plot of the speed of each method, in seconds, for dimensions up to 100.

available by request.

6. Comparison of Algorithms

A comparison was done between the ratio-of-uniforms method involving an acceptance-rejection method on the two-dimensional region Ω, and the method involving an acceptance-rejection method on the marginal distribution of U, followed by a quick generation of V. For dimensions 4 to 100, 500 trials were conducted, and the median time needed to generate a vector for each dimension was compared between the two ratio-of-uniforms methods. The results clearly showed that the latter method was quicker and more efficient. Less computations are needed because we only need to generate U before checking the rejection condition and potentially starting over. While squeeze functions are commonly used for a quick acceptance, in this case the squeeze function approach did not optimize the ratio-of-uniforms algorithm. As dimension increased, the time necessary to generate a vector on the shell increased linearly, as expected.

Incrementing the dimension by one requires a constant amount of additional work to generate a random variable on $B_d(1)$.

The slope of the method involving an accept-reject on Ω is approximately 27.63% larger than the slope of the method involving an accept-reject on the marginal distribution of U. Since the latter method is clearly faster than the former method, we use the latter method when comparing the ratio-of-uniforms method against the other methods to generate from the unit spherical shell.

For each method, in dimensions 3 to 100, 500 trials were conducted to get the average time needed to generate a uniformly distributed random variable on the shell.

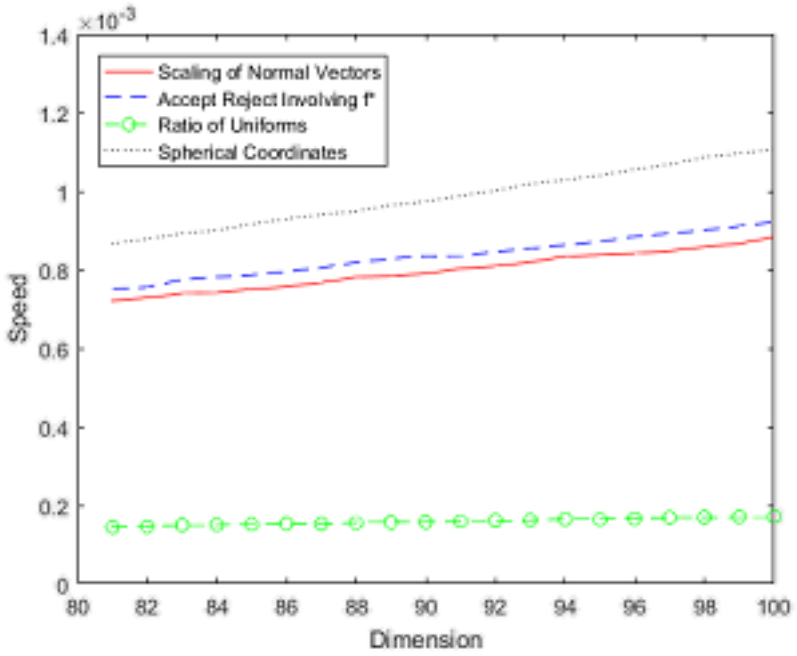

Figure 4. Plot of a close-up of the speed of the fastest four methods, in seconds, for dimensions 81 to 100.

See Figure 3 for the median time to generate one random variable for dimension up to 100. We also provide a close-up of the high dimensions to show that the ratio-of-uniforms clearly performs the best in both high and low dimensions (see Figure 4). It is clear from Figure 3 that the Gamma and Beta methods are the slowest, so we excluded them from the close-up. For each method, as the dimension increased, the time necessary to generate a vector on the spherical shell increased linearly.

The Beta and Gamma methods are the slowest methods to generate random variables from the unit spherical shell, largely because the generation of these random variables is complex and time-consuming. Clearly, the ratio-of-uniforms is the best choice for generating from the unit spherical shell. Since the ratio-of-uniforms method is not defined for $d = 3$, and the inverse transform method utilizes a simple computation, the best combination is to use the inverse transform method for $d < 4$ and the ratio-of-uniforms elsewhere. Compared to the other algorithms, the ratio-of-uniforms does not require as many operations. The slope for the scaling of normal vectors method, the method that is in second place, is over five times larger than the slope of the ratio-of-uniforms method. It is important to keep in mind that these differences are only in the order of 10^{-3} seconds. If one only needs to generate one random variable in a small dimension, these differences will be largely insignificant. However, with the availability of high-speed computers, people are tackling problems in high dimensions, and there is a new need to generate random variables in large dimensions. See Cai, Fan, & Jiang (2013), Soize (2008), Jiang (2008) for examples of problems being solved in high dimensions.

7. Conclusions and Extensions

We provide different algorithms to generate uniform random variates on the unit spherical shell. All of the alternatives presented here are based on recursive generation of random variates from the marginal distribution, being of particular interest the ratio-of-uniforms method presented in section 4.5.

The ideas presented here can be the basis to build a multidimensional extension of the Box-Muller method for isotropic densities (not only for the multivariate normal but also for the multivariate Student's t distribution). Also, our algorithm can be used to extend the algorithms based on the properties of spacings to odd dimensions.

Acknowledgement

We express our sincere gratitude to Stephanie Nagel (Hofstra University), Torin Quinlivan and Faustina Asante (Northern Illinois University), for their invaluable contributions to this research paper. Their dedication and hard work have significantly enriched the quality of the study, showcasing their commitment to academic excellence.

Disclosure statement

No potential competing interests.

Funding

No funding was received.

References

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Burr, I.W. (1942). Cumulative frequency functions. *Annals of Mathematical Statistics*, *13*, 215-232.

Burr, I.W., & Cislak, P.J. (1968). On a general system of distributions. i. the curve-shape characteristics. ii. the sample median. *Journal of the American Statistical Association*, *63*, 627–635.

Burr, I.W. (1973). Parameters for a general system of distributions to match a grid of a3 and a4. *Communications in Statistics*, *2*, 1-21.

Boisbunon, A. (2011). *The class of multivariate spherically symmetric distributions.* http://aurelie.boisbunon.free.fr/downloads/loisSS.pdf

Cheng, R.C.H. (1978). Generating beta variates with nonintegral shape parameters. *Communications of the ACM*, *21*, 317-322.

Cai, T., Fan, J., & Jiang, T. (2013). Distributions of Angles in Random Packing on Spheres. *Journal of Machine Learning Research*, *14*, 1837-1864.

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

Frenkel, D., & Smit, B. (2002). Understanding Molecular Simulation: From Algorithms to Applications (2nd ed.). Academic Press.

Hicks, J.S., & Wheeling, R.F. (1959). An efficient method for generating uniformly distributed points on the surface of an n-dimensional sphere. *Communications of the ACM*, *2*, 17-19.

Johnson, N.L., Kotz, S., & Balakrishnan, N. (1994). *Continuous Univariate Distributions*, Second ed, vol. 1. New York: John Wiley & Sons, Inc.

Jiang, W., & Tanner, M. A. (2008). Gibbs Posterior for Variable Selection in High-dimensional Classification and Data Mining. *The Annals of Statistics*, *36*(5), 2207-2231.

Kinderman, AJ., & Monahan, J.F. (1977). Computer generation of random variables using the ratio of uniform deviates. *ACM Transactions on Mathematical Software*, *3*, 257-260.

Muller, M. E. (1959). A note on a method for generating points uniformly on N-dimensional spheres. *Communications of the ACM*, 2(4), 19–20.

Luengo, E. (2022). Gamma Pseudo Random Number Generators.*ACM Computing Surveys*. 55(4),1-33. https://doi.org/10.1145/3527157

Mardia, K.V. and Jupp, P.E. (2000) Directional Statistics. John Wiley & Sons, London.

Marsaglia, G. (1972). Choosing a Point from the Surface of a Sphere. *The Annals of Mathematical Statistics*, 43(2), 645–646.

Marsaglia, G., & Tsang, WW. (2000). The Ziggurat Method for Generating Random Variables. *Journal of Statistical Software*.

Martinez, W.L., & Martinez, A.R. (2008). *Computational Statistics Handbook with MATLAB*, Second edn. Chapman & Hall/CRC.

Pharr, M., Jakob, W., & Humphreys, G. (2016). Physically Based Rendering: From Theory to Implementation (3rd ed.). Morgan Kaufmann Publishers.

Perez, C. J., Martín, J., Rojano, C., & Girón, F. J. (2008). Efficient generation of random vectors by using the ratio-of-uniforms method with ellipsoidal envelopes.*Statistics and Computing*, *18*, 209-217.

Soize, C. (2008). Construction of probability distributions in high dimension using the maximum entropy principle: Applications to stochastic processes, random fields and random matrices. *International Journal for Numerical Methods in Engineering*. Wiley. *76* (10), 1583-1611.