1 Overview

The aim of this work is two-fold: first, to give an overview of the Ziggurat algorithm Marsaglia and Tsang [undated] for unimodal absolutely continuous distributions; second, to provide one implementation if the algorithm. One of the beauties of the Ziggurat algorithm lies in the fact that while almost all the time it is a very efficient version of the classical rejection method, it allows to handle densities with infinite support—assuming it is known how to sample from the tail (in the normal case see, e.g., Marsaglia [1964]).

2 Formal Setup

In what follows, let f denote the density function; θ denote the mode of the distribution (argument where f achieves its maximum); and T denote its survival function:

$$T(x) = \int_x^\infty f(t)\,dt.$$

We will start with the monotone decreasing density case; the monotone increasing case can be handled similarly.

The idea behind the Ziggurat algorithm is to cover the density function with $n \geqslant 2$ horizontal layers of the same area, where all the layers except for the bottom one are rectangular. More formally, we start with a partition of the vertical interval

$$0 = f_0 < f_1 < \cdots < f_{n-1} < f_n = f(\theta).$$

For $1 \leqslant k \leqslant n-1$ let

$$a_k = \inf\{x : f(x) > f_k\}, \quad b_k = \sup\{x : f(x) > f_k\};$$

put $a_n = b_n = \theta$, and define the layers by

$$\begin{cases} L_0 = \{(x,y) : 0 \leqslant y \leqslant \min\{f_1, f(x)\}\}, \\ L_k = [a_k, b_k] \times [f_k, f_{k+1}], \quad 1 \leqslant k \leqslant n-1. \end{cases}$$

This layering is illustrated in Figure 1.

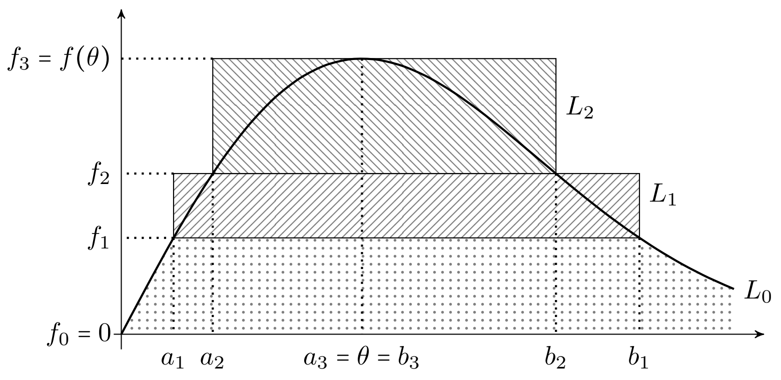

Figure 1: One-sided unimodal Ziggurat algorithm with 3 layers.

We want to choose $f_1, \cdots, f_{n-1}$ in such a way as to make sure that the area of each layer is the same:

$$|L_0| = |L_1| = \cdots = |L_{n-1}| = V.$$

Thus, one arrives at a (non-linear) system of n equations with n unknowns:

$$\begin{cases} V = (1 - T(a_1)) + (b_1 - a_1)\,f_1 + T(b_1) \\ V = (b_k - a_k)\,(f_{k+1} - f_k), \quad 1 \leqslant k \leqslant n-1. \end{cases} \tag{1}$$

Proposition 1. *System* (1) *has a unique solution.*

Having set up the basics, let us review the principles of the Ziggurat algorithm.

- The first step is to select one layer (uniformly) at random.
- The second step is to generate a (uniform) random point inside the selected layer. If the point happens to be under the graph of the density function, we accept it. Otherwise, we start over from the first step.

To facilitate these steps, note that every layer has a rectangular subset that lies entirely under the density curve. Specifically, all

$$B_k = [a_{k+1}, b_{k+1}] \times [f_k, f_{k+1}] \subseteq L_k, \quad 0 \leqslant k \leqslant n-1,$$

lie under the graph of f (for the top layer the box B_{n-1} is trivial). In light of this, the second step of the algorithm becomes substantially simpler if the point sampled from layer L_k falls inside B_k. The probability of this happening, which we will refer to as the *simple coverage probability*, is

$$p_k = \frac{|B_k|}{|L_k|}.$$

It is convenient to write these probabilities in terms of the widths of the layers (the term "width" is accurate for all layers but the bottom one, in which case we interpret it as the width of a rectangle of the same height and area). Let

$$a_0 = a_1 - (1 - T(a_1))/f_1 \quad \text{and} \quad b_0 = b_1 + T(b_1)/f_1,$$

and set

$$\begin{cases} w_k = b_k - a_k, & 0 \leqslant k \leqslant n-1, \\ w_n = 0. \end{cases} \tag{2}$$

Then the simple coverage probabilities become

$$p_k = \frac{w_{k+1}}{w_k} \quad \text{for } 0 \leqslant k \leqslant n-1. \tag{3}$$

Furthermore, let α_k and β_k, $0 \leqslant k \leqslant n-1$, denote the probabilities of landing to the left and right of B_k, respectively. Then

$$\begin{cases} \alpha_k = (a_{k+1} - a_k)/w_k, \\ \beta_k = (b_k - b_{k+1})/w_k, \end{cases} \quad \text{for } 0 \leqslant k \leqslant n-1. \tag{4}$$

Note, that in the case of monotone densities, either all $\alpha_k = 0$ or all $\beta_k = 0$, $0 \leqslant k \leqslant n-1$. It is not difficult to see that

$$\Pr\big(\text{simple coverage}\big) = \frac{1}{n}\sum_{k=0}^{n-1} \Pr\big(\text{simple coverage} \mid \text{layer } k\big) = \frac{p_0 + \cdots + p_{n-1}}{n},$$

$$\Pr\big(\text{rejection}\big) = \frac{1}{n}\sum_{k=0}^{n-1} \Pr\big(\text{rejection} \mid \text{layer } k\big) = 1 - \frac{1}{nV}.$$

3 Ziggurat Algorithm

The algorithm of Marsaglia and Tsang [undated] and Doornik [1997] for $n \geqslant 2$ layers is summarized below.

Algorithm (Continuous generators).

1. Generate $K \sim \text{Uniform}\{0, 1, \cdots, n-1\}$, the zero-based layer index.
2. Generate $U \sim \text{Uniform}[0, 1)$, the relative "horizontal position" inside L_K.

3. If $K = 0$:
    (a) If $U < \boxed{\alpha_0}$, accept a variate from the left tail.
    (b) If $U \geqslant \boxed{1 - \beta_0}$, accept a variate from the right tail.
    (c) If $\boxed{\alpha_0} \leqslant U < \boxed{1 - \beta_0}$, accept $\boxed{a_0} + U \cdot \boxed{w_0}$, since the random point landed inside B_0.
4. If $K \neq 0$:
    (a) Let $X = \boxed{a_K} + U \cdot \boxed{w_K}$ be the abscissa of the random point inside L_K.
    (b) If $\boxed{\alpha_K} \leqslant U < \boxed{1 - \beta_K}$, accept X, since the point landed inside B_K.
    (c) Generate $W \sim \text{Uniform}[0, 1)$, the relative vertical position inside L_K.
    (d) If $\boxed{f_K} + W \cdot \boxed{(f_{K+1} - f_K)} < f(X)$, accept X, since the point landed under the graph of f.
    (e) Return to step 1.

The "boxed" $\boxed{\text{quantities}}$ in the algorithm above can be pre-computed to speed up calculations.

In practice one usually has to deal with discrete uniform integer generators. For simplicity of presentation, we will assume that the possible values are $\{0, 1, \cdots, M\}$; typically, M would be a Mersenne number. The previous algorithm can be adjusted to take advantage of integer arithmetic in the following way:

Algorithm (Discrete uniform integer generators).

1. Generate $K \sim \text{Uniform}\{0, 1, \cdots, n-1\}$, the zero-based layer index.
2. Generate $\hat{U} \sim \text{Uniform}\{0, 1, \cdots, M\}$, the up-scaled relative "horizontal position" inside L_K.
3. If $K = 0$:
    (a) If $\hat{U} < \boxed{[\![(M+1)\alpha_0]\!]}$, accept a variate from the left tail.
    (b) If $\hat{U} \geqslant \boxed{[\![(M+1)(1-\beta_0)]\!]}$, accept a variate from the right tail.
    (c) If $\boxed{[\![(M+1)\alpha_0]\!]} \leqslant \hat{U} < \boxed{[\![(M+1)(1-\beta_0)]\!]}$, accept $\boxed{a_0} + \hat{U} \cdot \boxed{w_0/(M+1)}$, since the random point landed inside B_0.
4. If $K \neq 0$:
    (a) Let $X = \boxed{a_K} + \hat{U} \cdot \boxed{w_K/(M+1)}$ be the abscissa of the random point inside L_K.
    (b) If $\boxed{[\![(M+1)\alpha_K]\!]} \leqslant \hat{U} < \boxed{[\![(M+1)(1-\beta_K)]\!]}$, accept X, since the point landed inside B_K.
    (c) Generate $\hat{W} \sim \text{Uniform}\{0, 1, \cdots, M\}$, the up-scaled relative vertical position inside L_K.
    (d) If $\boxed{f_K} + \hat{W} \cdot \boxed{(f_{K+1} - f_K)/(M+1)} < f(X)$, accept X, since the point landed under the graph of f.
    (e) Return to step 1.

As before, the "boxed" $\boxed{\text{quantities}}$ in the discrete version of the algorithm can be pre-computed; moreover, some of them are now integer-valued. Note, that if we know in advance that $\alpha_0 = \cdots = \alpha_{n-1} = 0$ or $\beta_0 = \cdots = \beta_{n-1} = 0$, certain steps in the algorithm can be simplified (or skipped entirely).

Remark 1. In most cases, we think it is reasonable to make the following assumptions.

- If $\alpha_j = 0$ for some j, then all $\alpha_k = 0$, $0 \leqslant k \leqslant n-1$.
- If $\beta_j = 0$ for some j, then all $\beta_k = 0$, $0 \leqslant k \leqslant n-1$.

Remark 2. Also note, that only $\lceil \log_2(n) \rceil$ bits are necessary to generate K; the remaining bits may be used, e.g., to implement vectorized Ziggurat versions, or store the random index for repeated iterations.

Remark 3. A couple of words on probability re-scaling, discrete generators, and rounding up or down. Suppose M is a positive integer, $Y \sim \text{Uniform}[0,1)$, and $X = \lfloor (M+1)Y \rfloor \sim \text{Uniform}\{0,1,\cdots,M\}$. Our goal is to approximate the events $\{Y < p\}$ or $\{Y \geqslant p\}$ with events generated by X.

To that end, first suppose $p = k/(M+1)$ for some integer k, $0 \leqslant k \leqslant M+1$. Then

$$\{Y < p\} = \{X < k\} \qquad \text{and} \qquad \{Y \geqslant p\} = \{X \geqslant k\}.$$

In general, if $0 \leqslant p \leqslant 1$, we want to round it toward the nearest multiple of $1/(M+1)$. Note, that in some cases $(M+1)$ would result in an overflow, so we want to handle the case when $p \geqslant (M+0.5)/(M+1)$ separately.

References

Jurgen A. Doornik. An improved ziggurat method to generate normal random samples. *Communications in Statistics – Theory and Methods*, 26(5):1253–1268, December 1997.

George Marsaglia. Generating a variable from the tail of the normal distribution. *Technometrics*, 6(1): 101–102, February 1964. ISSN 00401706.

George Marsaglia and Wai Wan Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software*, 5(i08), undated.