

The Ziggurat Method

Daniel de Souza Severo

University of Toronto

danielsouzasevero@gmail.com

July, 2014

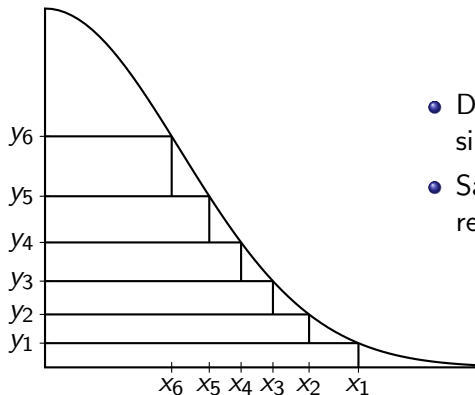
Overview

- 1 Introduction
- 2 First Glance
- 3 Region Sampling
- 4 Method
- 5 Implementation and Speed Comparison
- 6 References
- 7 Appendix

This method was originally created by Marsaglia [3]. It has been improved by many including McFarland [2]. This presentation is based on both of these papers.

- Pseudo-random number generators (PRNGs) are crucial in the context of simulating noise in communication channels.
- We present a report on an efficient method for generating pseudo-random samples from any decreasing probability distribution called the **Ziggurat Method**.

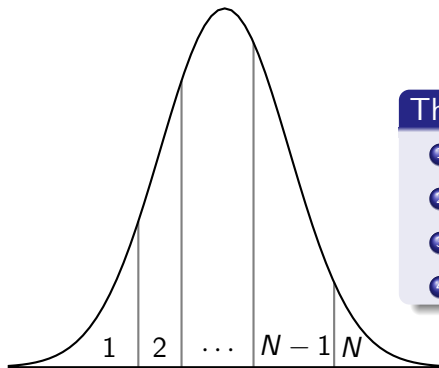
First Glance



- Divide the distribution into simpler regions.
- Sample proportional to each region.

Region Sampling (1/2)

Given a complicated distribution g that encloses an area A , how can we create a dataset with distribution g without sampling directly from the density itself?



Theorem

- 1 Divide A into $\{A_1, \dots, A_N\}$.
- 2 Select A_i with probability $\mu(A_i)$.
- 3 Uniformly sample (x, y) from A_i .
- 4 Return x .

Figure: A general distribution g being divided into N regions A_i with area $\mu(A_i)$.

Region Sampling (2/2)

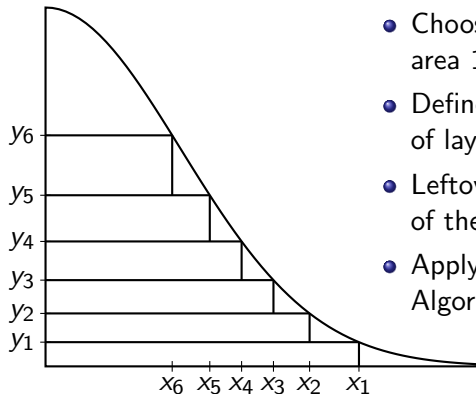
Given a complicated distribution g that encloses an area A , how can we create a dataset with distribution g without sampling directly from the density itself?

Theorem

- 1 *Divide the region A under g into subregions A_i such that $\{A_i, \dots, A_N\}$ is a partition^a of A .*
- 2 *Randomly select a region A_i with probability proportional to its area, $\mu(A_i)$.*
- 3 *Uniformly sample a point $p = (x, y)$ from the selected region A_i .*
- 4 *Return x , since it will have distribution g .*

^aThis means that $A = \bigcup_{i=1}^N A_i$ and $A_i \cap A_j = \emptyset$ for all $i \neq j$.

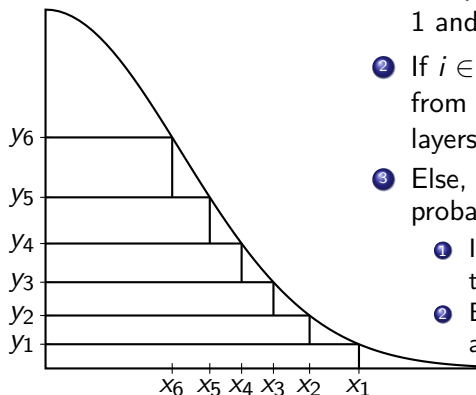
Method (1/2)



- Choose a number N of layers each with area $1/N$.
- Define L_{max} as the maximum number of layers that fit under g .
- Leftover regions represent $1 - L_{max}/N$ of the area under g .
- Apply the Rejection Sampling Algorithm.

Method (2/2)

Example for $N = 8$



- ❶ Sample an integer i uniformly between 1 and $N = 8$.
- ❷ If $i \in [1, L_{\max} = 6]$, return x uniformly from region $[0, x_i]$. (This represents the layers).
- ❸ Else, sample an integer $j \in [1, 7]$ with probability $p(j) = \mu(A_j)$.
 - ❶ If $j = 1$, return a sample x from the tail.
 - ❷ Else, apply Rejection Sampling to get a uniform sample x from region j .

Implementation and Speed Comparison

- The Ziggurat algorithm implemented in C for $N = 256$, is over 4 times faster than the tradition polar method [1], running on an Intel i7-4790 clocked at 3.60 GHz with 16 GB of RAM.
- McFarland [2] has made available all the necessary codes to be used in C, Python and MATLAB. It can be found at https://bitbucket.org/cdmcfarland/fast_prng.
- We have uploaded a Python script that generates all the necessary tables (x_i , $y_i = g(x_i)$ and A_j) for any given bin size N : https://github.com/dsevero/A-Report-on-the-Ziggurat-Method/blob/master/ziggurat/generate_tables.py

- 1 George Marsaglia, **A convenient method for generating normal variables.**, SIAM Rev. 6, 260-264, 1964.
- 2 Christopher D. McFarland, **A modified ziggurat algorithm for generating exponentially- and normally-distributed pseudorandom numbers.**, Apr. 2014.
- 3 George Marsaglia, Wai Wan Tsang, **A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions**, SIAM Journ. Scient. and Statis. Computing, 5, 349-359, 1984.

Appendix (1/3)

Let I be a random variable with distribution $f_I(i) = \mu(A_i)/\mu(A) = \mu(A_i)$. If $P = (X, Y)$ represents the points uniformly sampled in each region, then $f_{P|I}(p|i) = \mathbb{1}\{p \in A_i\}/\mu(A_i)$.¹ We can now calculate $f_P(p) = f_{X,Y}(x, y)$:

$$f_P(p) = \sum_i f_I(i) f_{P|I}(p|i) = \sum_i \mu(A_i) \frac{\mathbb{1}\{p \in A_i\}}{\mu(A_i)} = \sum_i \mathbb{1}\{p \in A_i\}$$

A specific point p can only belong to one subregion, since $A_i \cap A_j = \emptyset, \forall i \neq j$. Hence, we have that:

$$f_P(p) = \sum_i \mathbb{1}\{p \in A_i\} = \mathbb{1}\{p \in A_1\} + \mathbb{1}\{p \in A_2\} + \cdots + \mathbb{1}\{p \in A_N\} = 1$$

$$f_X(x) = \int_y f_{X,Y}(x, y) dy = \int_0^{g(x)} dy = g(x)$$

¹ $\mathbb{1}\{x\} = 1$ if x is true and 0 if it is false.

Appendix (2/3)

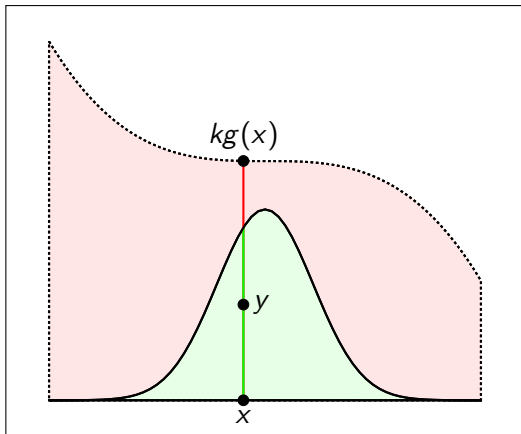


Figure: RSA rejection/acceptance regions. After sampling x from g (.....) if y falls under curve h (green) then it is stored, otherwise if it is above h (red) it is rejected.

Appendix (3/3)

Define $X \sim f_X(x) \equiv g(x)$ and $Y \sim f_{Y|X}(y|x) = \text{Uniform}[0, kg(x)]$. Now, define $Z = \mathbb{1}\{Y < h(X)\}$ such that if the sample y is under the curve h , then Z takes on the value 1, otherwise 0.

$$f_{Z|X}(z = 1|x) = \frac{h(x)}{kg(x)}$$

Consider now the pair (X, Z) where $(X, Z = 1)$ represents the sample points X that we will keep according to the RSA.

$$f_{X,Z}(x, z = 1) = f_X(x)f_{Z|X}(z = 1|x) = g(x)\frac{h(x)}{kg(x)} = \frac{1}{k}h(x)$$