A Random Walk to Capacitance

Reyhaneh Bahranifard
reyhaneh.bahranifard@ontariotechu.net

Abstract

This project looks at methods for generating uniform point distributions on the surfaces of convex objects, focusing on the isotropic random walk. The random walk method is used to sample points on the surface of a convex body, such as a cube or sphere, by repeating steps determined by random directions and the intersection with the object's surface. The goal is to estimate capacitance by studying the behavior of the sampled points. The approach's effectiveness is measured by calculating the capacitance of a cube and analyzing the uncertainty in the estimates based on the number of sampled points.(3)

1 Random Points on a Sphere

1.1 Muller's Method

In this method, independent random samples are generated using a normal distribution. After that, the samples are normalized to ensure that the points lie on a sphere.(2)
To generate Gaussian random samples from computer random generators, the following conversion equations are used. The reasoning for this in 2-D space is explained in a repository on my GitHub profile at this link:

$$Y_1 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

$$Y_2 = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

where U_1 and U_2 are uniform variables and Y_1 and Y_2 are Gaussian random variables. To extend this to N dimensions, the conversion can be applied on pairs like $(U_1, U_2), (U_3, U_4), \ldots$.
Normalization ensures that the point lies on the surface of a unit sphere, meaning the distance from the origin is exactly 1. To normalize a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, each component of the vector is divided by its norm. Mathematically, the normalized vector $\hat{x}$ is given by:

$$\hat{x} = \frac{x}{\|x\|}$$

After normalization, the new vector $\hat{x}$ satisfies:

$$\|\hat{x}\| = 1$$

This holds true because:

$$\|\hat{x}\| = \sqrt{\hat{x}_1^2 + \hat{x}_2^2 + \cdots + \hat{x}_n^2}$$

Substituting $\hat{x}_i = \frac{x_i}{\|x\|}$:

$$\|\hat{x}\| = \sqrt{\left(\frac{x_1}{\|x\|}\right)^2 + \left(\frac{x_2}{\|x\|}\right)^2 + \cdots + \left(\frac{x_n}{\|x\|}\right)^2} \tag{1}$$

$$= \frac{1}{\|x\|}\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \frac{1}{\|x\|}\|x\| = 1$$

Therefore, after generating Gaussian random variables, equation (1) should be calculated.

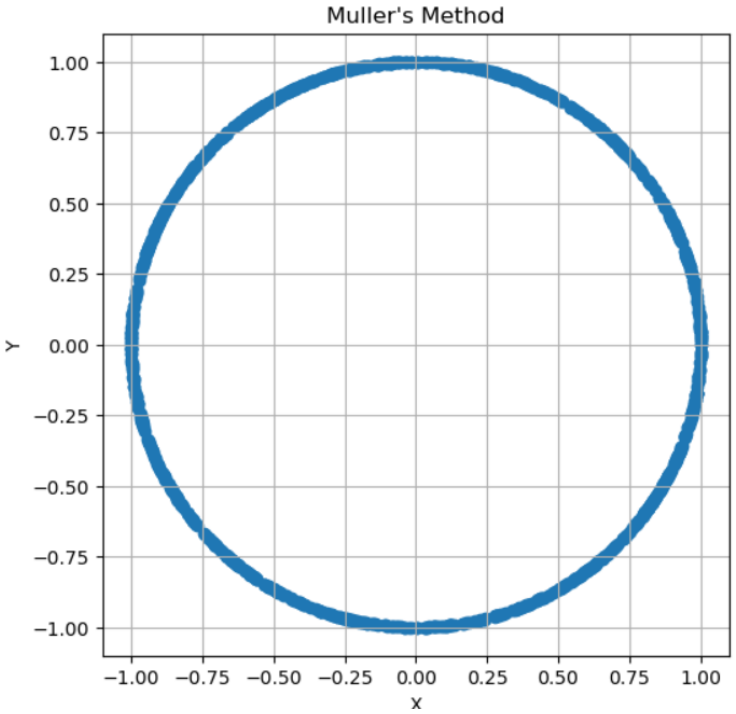

Figure 1: Visualization of the Muller method in 2D, with 1000 samples

1.2 Rejection Method

Rejection sampling is a method used to generate random samples from a target distribution $f(x)$ when direct sampling is difficult, and a simpler proposal distribution $g(x)$ is utilized to facilitate the process.The acceptance rate is defined as:

$$\alpha = \frac{f(x)}{M \cdot g(x)}$$

A candidate x is accepted with probability α by comparing it to a uniformly distributed random number $U \sim \text{uniform}(0, 1)$. If $U \leq \alpha$, the sample is accepted; otherwise, it is rejected, and the process is repeated.

For this specific case:

1. A random vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is generated, where each x_i is sampled from a uniform distribution over the interval $[-1, 1]$.

2. The squared Euclidean norm of the sampled vector is computed:

$$\|\mathbf{x}\|^2 = x_1^2 + x_2^2 + \cdots + x_n^2.$$

3. If $\|\mathbf{x}\|^2 > 1$, the sample is rejected, and a new vector is generated.

4. If $\|\mathbf{x}\|^2 \leq 1$, the vector is normalized so that it lies on the unit sphere:

$$\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

where

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

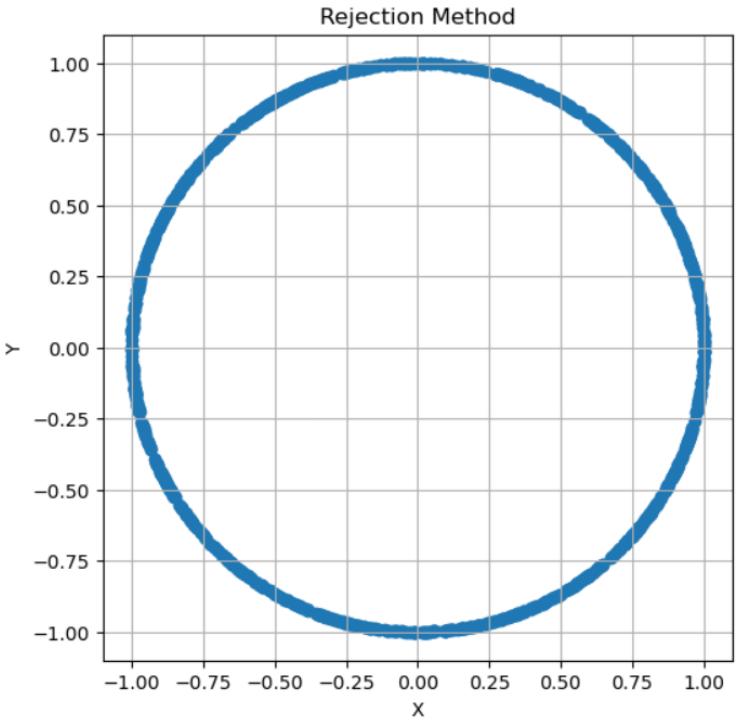

Figure 2: Visualization of the Rejection method in 2D, with 1000 samples.

1.3 Comparison

In this section, a basic comparison between the Muller method and the rejection method is provided.

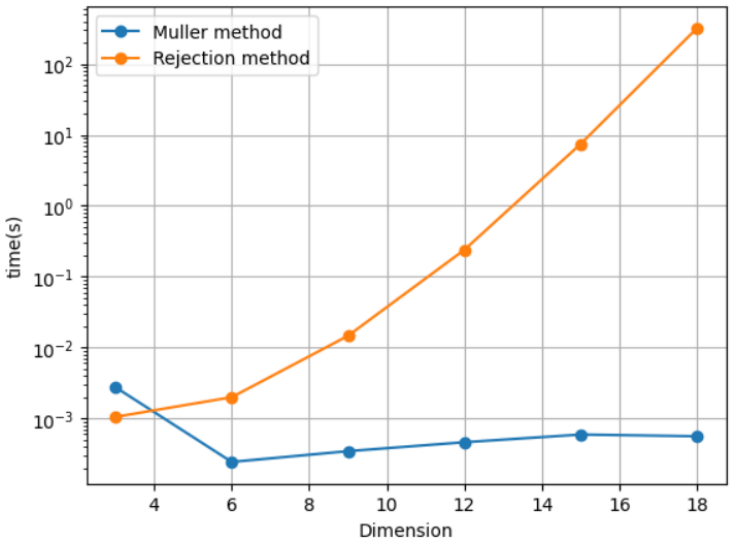

Figure 3: Comparison between two methods in different dimensions,number of samples=1000

When the dimension increases, the rejection method becomes progressively worse. This can be attributed to several reasons. First, for a normalized cube, the volume can be expressed as $V_{\text{cube}} = 2^N$. As N increases, the number of required points grows exponentially. For instance, in 10 dimensions, 400,000 points are needed while many of which will be rejected. On the other hand, there is no rejection condition in the Muller method, which speeds up the process since random points are generated directly within a sphere.

2 Isotropic Random Walk on a Convex Surface

2.1 Implementation

The Isotropic Random Walk on a Convex Surface is a method used to generate points that are uniformly distributed on the surface of a convex object, such as a cube.

Algorithm 0.1 Isotropic Random Walk on a Convex Surface

Initialization: Let G be a convex object. Starting at a point $\boldsymbol{x} \in G$, generate a direction $\boldsymbol{d}_0$ from the uniform distribution of directions, and walk in direction $\boldsymbol{d}_0$ until you hit the surface of G. Call the resulting point $\boldsymbol{y}_0$.

for $j = 1, \ldots, p-1$

Generate a direction $\boldsymbol{d}_j$ from the uniform distribution of directions, and walk from $\boldsymbol{y}_{j-1}$ in direction $\pm\boldsymbol{d}_j$ until you hit another point on the surface of G. Call the resulting point $\boldsymbol{y}_j$.

end

Figure 4: Isotropic Random Walk on a Convex Surface

The algorithm begins at an initial point on the surface and generates a random direction for each step. The direction of movement is generated using the Muller method. The walker moves in the generated direction until it intersects the surface again, at which point the new position becomes the next step. This process continues for a desired number of steps.
To summarize:

1. Start with an initial point, for example, $(1, 0, 0)$.
2. Apply the Muller method to determine the desired direction of movement, for instance, the result is (d_x, d_y, d_z). The direction is a vector, which also has a magnitude that indicates how fast the walker will move.
3. Define α in each direction to determine how far the random walker must go to reach the surface again:

$$\alpha_i = \frac{\text{boundary position} - \text{current position}}{d_i}$$

4. Choose the minimum α, which indicates that the walker should move in the direction where the surface is reached with the least movement. Then calculate the new position:

$$\text{new position} = \text{current position} + \alpha \times \text{direction}$$

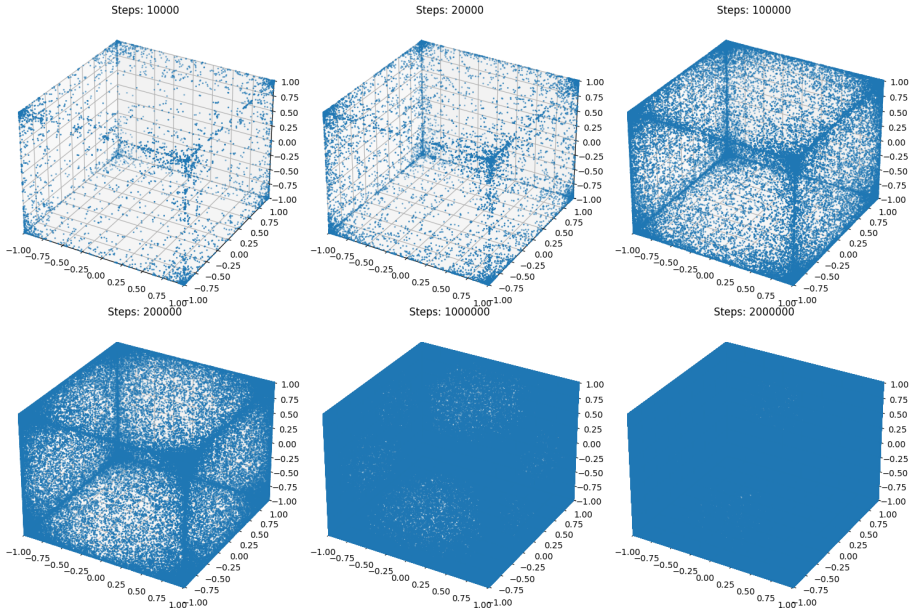

Figure 5: The algorithm was run for different numbers of steps: $[10^4, 2 \times 10^4, 10^5, 2 \times 10^5, 10^6, 2 \times 10^6]$. The accumulation of points is slightly higher in the corners and, to a lesser extent, along the edges, which is natural, as these locations are intersections where the probability of placement is higher.

2.2 Challenges

There are several important points to consider for this algorithm:

- **Floating-Point Precision**: An example is whether the point $p = [1.00001, 0, 0]$ is on the surface or not. The answer is yes, and it is interpreted as a floating-point error. To handle floating-point numbers, coordinate clipping was used as a solution. For example, if $p = [0.999, 0.1, -0.2]$, it is transferred to $[1.0, 0.1, -0.2]$ by clipping the coordinate.

- **Bias Prevention**: Uniform sampling was ensured by using the Muller method for direction generation. Also, only odd-indexed points (i % 2 == 1) were stored to reduce correlation in the random walk. More details on this will be provided in the next part.

- **Tangent Movement Mitigation**: Tangent movements were avoided by discarding invalid steps (where direction[i] != 0) and enforcing face intersections using scaled directional steps (alpha * direction).

2.3 Uniform Point Distribution

The direction of the random walker was chosen using the Muller method at each time step to ensure that the directions are chosen without any bias, and the movement is isotropic. In this section, the accuracy of this statement was tested.

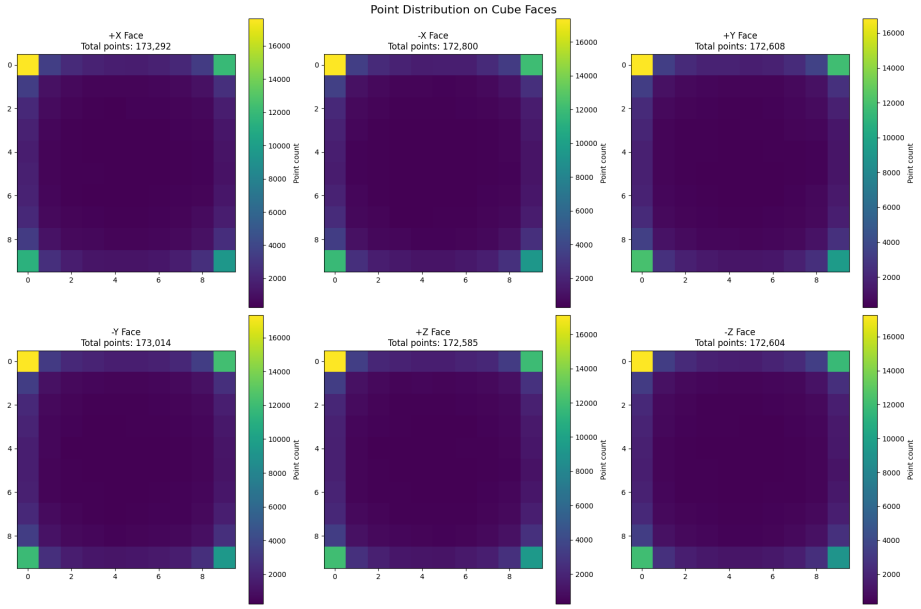

Figure 6: Each face of the cube was divided into 100 squares, and the number of points was calculated and written at the top of each heatmap. The total number of points is 2×10^6.

As shown in Figure 6, the distribution of points is almost uniform, with higher values in the corners and along the edges. As mentioned, this is natural because these locations are intersections where the probability of placement is higher. Moreover, the total number of points on each face shows that they are approximately equal.

3 Application: Capacitance on a Convex Surface

One application of the random walk on a convex surface is the calculation of capacitance, a physical quantity, on a surface.

The following formula was used to calculate capacitance using generated points:(1)

$$C = \left(\lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} \nu(y_n) \right)^{-1},$$

where

$$\nu(y) = \frac{1}{|x - y|}. \tag{2}$$

Here, x is any point inside the domain G, and y is on the boundary ∂G. The process is explained step by step below:

1. For a cube centered at the origin:

$$\mathbf{x} = [0, 0, 0]^T \quad \text{(center)}.$$

2. Random points $\mathbf{y}_n$ are generated on the cube.

3. For each $\mathbf{y}_n = [y_1, y_2, y_3]^T$:

$$|\mathbf{x} - \mathbf{y}_n| = \sqrt{(0 - y_1)^2 + (0 - y_2)^2 + (0 - y_3)^2} = \sqrt{y_1^2 + y_2^2 + y_3^2}.$$

4. Equation (2) is used to calculate $\nu(\mathbf{y}_n)$.

5. The capacitance is approximated as:

$$C \approx \left(\frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathbf{x} - \mathbf{y}_n|} \right)^{-1}.$$

The theoretical capacitance value for a cube with a side length of 2 is considered to be $C = (1.33266 \pm 0.08844)$.(4) Two key patterns are revealed by the

Number of Points	**Mean Estimate**	**Theoretical Value**	**Standard Deviation**
10^7	1.46640 ± 0.00019	1.33266	0.00019
10^6	1.46650 ± 0.00057	1.33266	0.00057

Table 1: Capacitance estimation results for different numbers of points. Standard deviation is calculated using 10 trails for each number of points

results. First, a consistent 10% overestimation is observed across all sample sizes, indicating that this bias is caused by systematic errors rather than statistical limitations. When the number of points is increased from 10^6 to 10^7, the standard error is reduced by $3\times$ ($0.00057 \to 0.00019$), but the relative error remains fixed at $\approx 10\%$. This confirms that the primary source of error is not related to sampling variance. The persistent deviation is primarily caused by numerical round-off errors near the cube corners and discretization effects from the random walk's finite step size in high-curvature regions. It is worth

mentioning that estimating the actual value for the capacitance of a cube is not a well-defined problem and is almost unsolvable. The random walk approach allows for the accurate sampling of points on the boundary of the cube, thereby avoiding issues associated with sharp potential gradients and numerical artifacts near the corners. This makes the random walk method a powerful tool for capacitance calculations, especially in complex geometries where other techniques may struggle due to discretization or numerical errors.(1)

References

[1] Hwang, C.O., Mascagni, M., Won, T.: Monte Carlo methods for computing the capacitance of the unit cube . https://doi.org/https://doi.org/10.1016/j.matcom.2008.03.003

[2] MULLER, M.E.: A Note on a Uniformly Method for Generating Points on N-Dimensional Spheres (1959). https://doi.org/https://doi.org/10.1145/377939.377946

[3] O'Leary, D.P.: A Random Walk to Capacitance `https://www.cs.umd.edu/%7Eoleary/SCCS/supp/random_capac/rcapac.pdf`

[4] Reitan, D.K., Higgins, T.J.: Calculation of the Electrical Capacitance of a Cube . https://doi.org/https://doi.org/10.1063/1.1699929