A Random Walk to Capacitance

Dianne P. O'Leary

My high school physics teacher told us that we didn't have the 'capacicity' to understand the subject, since we hadn't had calculus. In this case study, we'll see if we have the capacity to estimate capacitance.

The **capacitance** of an object determines its capacity to store electrical charge. For example, the undersides of clouds can accumulate extra electrons, causing a negative charge. This induces a positive charge on the ground below. When the build-up becomes too great, the capacitor discharges, forming lightning. Capacitance has many practical uses; for example, computer memory (RAM) relies on capacitance to store data. Knowing the capacity of an object for charge build-up is essential in using that object to design useful devices.

The capacitance of an object depends on its shape, its size, and its material properties. For example, the capacitance of a sphere is proportional to its radius, and the closed-form formula is well known. Formulas are also known for a cylinders and for a pair of parallel plates. The cube, though, is a shape for which no formula is known.

In this case study we will develop and use an algorithm to estimate the capacitance of a cube using a random walk on the surface of the cube. The algorithm works for any convex object.[1]

Random Walks on Surfaces

In order to take a random walk on the surface of an object, we will need to generate points that are uniformly distributed on the surface of a sphere. We evaluate two methods for doing this in the next two challenges.

[1] An object is convex if the line segment connecting any two points in the object stays within the object.

POINTER 0.1. Random Points on a Sphere
In this case study we consider two interesting algorithms for generating points that are uniformly distributed on the surface of a sphere:

- One method can be found in the paper: Muller, M. E. "A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres," *Comm. Assoc. Comput. Mach.* 2, pp. 19-20, Apr. 1959.
- The **rejection method** generates points uniformly distributed in a cube containing the sphere, rejects any that are outside the sphere, and scales the others to lie on the sphere's surface.

Both of these methods work for 2-d, 3-d, 4-d, etc. Another method, the **trig method**, works only in 3-d.

What we are calling **uniformly distributed points on the surface of the sphere**, or **uniformly distributed directions**, are sometimes called **uniformly (isotropically) distributed points within the solid angle** with vertex equal to the previous point.

CHALLENGE 0.1. Explain why the rejection method, defined in Pointer 0.1, works.

CHALLENGE 0.2. Use Muller's method to generate 1000 points uniformly distributed on the surface of a sphere, using `randn`. Then use the rejection method to do the same thing, using `rand`. Make both methods run as fast as you can. If the random number generators are good enough (and I believe that they are), then the two sequences should both have good "randomness" properties, so the methods can be judged solely on speed. Write a short description of the methods, your experiment, the results, and your recommendation of which method to use.

The capacitance of a sphere is well understood, so why do we need to generate random points on a sphere? It turns out that this will enable us to do a random walk on the surface of any convex object, including a cube. The reason is that a point on a sphere centered at the origin corresponds to a direction from the origin, and if we have a uniformly distributed set of points, we have a uniformly distributed set of walking directions.

POINTER 0.2. Pitfalls for Challenge 1

- Don't try to argue about the "number" of real numbers in a line segment or a wedge, since you are then arguing about infinity and get into paradoxes. For example, the "number" of points in a line segment is the same as that in a line segment of twice the length, since we can put them in a 1-to-1 correspondence.
- Don't try to argue about the number of floating-point numbers on a radius, either. Because the floating-point numbers form a grid, you can't argue that every radius of the sphere contains the same number of floating-point numbers.

Every (sufficiently large) wedge will contain approximately the same number of floating-point numbers as other wedges of the same size, though, so it is ok to base the argument on what happens in floating-point on a wedge.

POINTER 0.3. Pitfalls for Challenge 2

- Your documentation should give the reference to the Muller paper and also a reference for the rejection algorithm.
- It is a good idea to replicate your trials and compute a mean and standard deviation for your timings.

Let G be our convex object. Algorithm 0.1 is a way to generate p points on the surface of G, drawn from a particular distribution that will help us compute capacitance.

There is one important change that needs to be made to the algorithm whenever any part of the surface of the convex object is flat: we need to discard the iterates for $j = 2, 4, 6, \ldots$. To understand the reason, consider the cube. If we are on a particular face, the probability that our next point will be on that face is 0, because it is very unlikely that our direction will lie in the plane defined by the face. If we only look at the odd iterates, though, then a step from a particular point to any other small surface region has nonzero probability, and we need this property.

If the object is very symmetric, violating this property really wouldn't hurt the randomness of a long sequence of points, but we'll observe the discarding rule anyway.

Let's see how these points are distributed on the surface of 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

POINTER 0.4. Pitfalls for Challenge 3

- You need to use Muller's method (not uniform random numbers) and to walk from point-to-point, not from the center to the surface.
- Finding the intersection with the surface of the cube is a "simple" mathematical exercise, but a very complicated task in floating point. Is [1.00000000001, .5, .5] on the surface of the cube or outside of it? This is a very delicate issue that has to be carefully programmed.
- Mascagni and Simonov, and the monograph they cite by Sabelfeld and Simonov, claim that for a general convex body, there is always an intersection of the step with the surface of the body, but this does not seem to be true; we could, for example, generate a step tangent to the body and never intersect it.

CHALLENGE 0.3. Let G be the cube centered at $\boldsymbol{x} = [0, 0, 0]^T$ with a vertex at $[1, 1, 1]^T$. (Note that the length of each side of the cube is 2.)

Use Algorithm 0.1 for $p = 2 \times 10^6$. (Don't try to store all of these points and directions. Process them one at a time.)

Evaluate how uniform the resulting 10^6 points are by dividing each face of the cube into 100 squares and counting how many points fall within each square. Display your results graphically; use, for example, `imagesc` to display the 600 counts, and describe the results in a sentence.

POINTER 0.5. Choices
There are many ways to find the intersection of a line with the surface of the cube.

- You could solve a minimization problem: minimize the distance between the point $\boldsymbol{y} + \alpha \boldsymbol{d}$ and the surface, over all choices of the parameter α.
- You could solve a nonlinear equation: distance $= 0$.
- You could express it as a linear programming problem.
- You could compute the intersection of the line with each of the planes defined by the faces of the cube, and pick the intersection point that lies on (rather than outside) the face.

The last way in this list will be fastest.
To get comfortable with this task, you might want to draw the picture for 2-d first.

Charge Density and Capacitance

In order to compute capacitance, we need the following principle from physics, which you can take on faith:

$$\int_{\partial G} \frac{1}{\|\boldsymbol{x} - \boldsymbol{y}\|} \, \mu(\boldsymbol{y}) \, \mathrm{d}\sigma(\boldsymbol{y}) = 1.$$

In this equation,

- ∂G denotes the surface of the object G.
- $\boldsymbol{x}$ is a fixed point inside G. The equality holds for any choice of this point.
- $\boldsymbol{y}$ denotes a point on the surface of G.
- $\mu(\boldsymbol{y})$ is the (unknown) **charge density** at $\boldsymbol{y}$.
- $\mathrm{d}\sigma(\boldsymbol{y})$ denotes the infinitesimal surface area of G at $\boldsymbol{y}$.
- $\|\boldsymbol{z}\|^2 = \boldsymbol{z}^T \boldsymbol{z}$.

Let $v(\boldsymbol{y}) = 1/\|\boldsymbol{x} - \boldsymbol{y}\|$. Then the Integral Mean Value Theorem from calculus tells us that

$$\begin{aligned} 1 = \int_{\partial G} \frac{1}{\|\boldsymbol{x} - \boldsymbol{y}\|} \, \mu(\boldsymbol{y}) \, \mathrm{d}\sigma(\boldsymbol{y}) &= \int_{\partial G} v(\boldsymbol{y}) \, \mu(\boldsymbol{y}) \, \mathrm{d}\sigma(\boldsymbol{y}) \\ &= \bar{v} \int_{\partial G} \mu(\boldsymbol{y}) \, \mathrm{d}\sigma(\boldsymbol{y}) \\ &= \bar{v} \, C(G), \end{aligned}$$

POINTER 0.6. Pitfalls for Challenge 4

- Note that the average value of (1 / distance) is not the same as 1 / (average value of distance).
- Because of the important role played by edges and corners and their neighborhoods, I worry that the estimate of capacitance will not necessarily converge to the correct value as the number of sample points goes to infinity. I am particularly concerned about bad effects from round-off near the corners. The estimate depends on the number of digits carried in the floating-point numbers, and I am not sure that double precision is enough to guarantee more than a couple of correct digits. It would be interesting to run the random walk in quadruple precision and see what happens.
- Bottom line: Monte Carlo methods really are methods of last resort. They depend on having a very high quality pseudo-random number generator and, for random walks, guaranteeing that artifacts of floating-point arithmetic do not destroy the results. These are indeed very difficult issues!

---

where $\bar{v}$ is the average value of v on the surface and $C(G)$ is (by definition) the capacitance of G.[2]

We can estimate $\bar{v}$ by sampling points on the surface of the cube using the probability density function defined by $\mu(\boldsymbol{y})$, which is exactly what Algorithm 0.1 does. An estimate of $\bar{v}$ then gives us an estimate for $C(G)$.

CHALLENGE 0.4. Estimate the capacitance of the cube centered at the origin with a vertex at $[1,1,1]^T$ using 10^7 points.

(a) Estimate the uncertainty in your estimate of the capacitance. To do this, compute the variance in the 10 estimates obtained from each group of 10^6 points.

(b) Re-estimate the capacitance using 10^8 points. By what factor did your estimate of variance decrease? Is this consistent with Monte Carlo theory? (Note: if you can't run 10^8 points quickly enough, repeat using 10^6 points instead.)

---

[2] Capacitance has units of farads in the SI system, but we'll be a bit vague about how we have scaled the equation.

POINTER 0.7. Looking Ahead
Use this case study to push yourself toward computational independence. In the "real world" of research or development, you will need to find and evaluate relevant literature, derive useful mathematical properties of your problem, test algorithms, convince your coworkers and supervisors that your ideas are sound, implement your ideas in a well-documented computer program, and present your results in a way that other people can understand, all within assigned deadlines. The challenges in this case study are designed to give you practice in these tasks.

POINTER 0.8. Further Reading.
See Chapter 18 of the SCCS textbook for more applications of Monte Carlo integration.
The algorithm in this case study is taken from "The Random Walk on the Boundary Method for Calculating Capacitance," Michael Mascagni and Nikolai A. Simonov, *J. of Computational Physics* 195 (2004) 465–473. You do not need to read that article to solve the challenges.
For more information on capacitance (again, not necessary to solve this case study), see, for example `http://micro.magnet.fsu.edu/electromag/electricity/capacitance.html`
