Algorithms for localized dart-throwing

Richard Crandall
Perfectly Scientific, Inc.

Abstract. How does one "throw localized darts," that is, place random points in a subregion? It is inefficient just to find points in a larger region, only to check painstakingly every generated point for subregion membership. We describe herein several means for efficient, localized dart-throwing, under the constraint that a subregion's points be entirely consistent with a restriction of any larger generated point set to said subregion. The methods proposed herein work in any number of spatial dimensions. We define a formal Fourier power spectrum which for certain algorithm variants can be used to interpret the level of "randomness" for a dart-throwing scheme.

30 Mar 2000

1. Statement of the problem

The "localized dart-throwing" problem is described as follows. Given a region R for which there is a meaningful dart-throwing algorithm (i.e., method for generation of random points in R, under some specified randomness criterion), one wishes to throw darts into a subregion $S \subseteq R$. Yet, these darts falling in S are to follow precisely the pattern that would result by a simple restriction, to S, of the dart-throwing into R.

Various approaches described herein have to circumvent an obvious obstacle: a random-number generator is usually difficult to "advance," i.e., if a generator be throwing darts over the larger region R. it is generally unclear *a priori* how to find indices of those points that actually fall into S. This obstacle reminds us of such as discrete logarithm problems as occur in cryptographic scenarios, for example how does one find those powers y such that $x^y \bmod z$ that lie in a subinterval of $[1, z-1]$?

In what follows, let us assume D spatial dimensions, with

$$R = [0,1)^D,$$

being the unit D-cube, as the maximal region. Denote by $\{r_n\}$ a set of random (in some appropriate sense) vectors, $r_n \in R$. The problem then is to find, for a given $S \subseteq R$, those $r_n \in S$. If one desires a prescribed *density* of points, we simply say that an N be given, and for the region R the full dart set is taken to be

$$\{x_0, x_1, \ldots, x_{N-1}\}$$

so that there will be N points per unit volume. The goal in this case of a working algorithm would be to find the reduced point count of roughly

$$\lfloor N \frac{\mu(S)}{\mu(R)} \rfloor$$

points that fall into S, where μ denotes D-dimensional measure (generalized volume). This reduction to S can be thought of as the determination of the subset M of indices such that $x_n \in S$ if and only if $n \in M$.

2. Universal reduction to a 1-dimensional scenario

We can reduce the stated problem to a 1-dimensional setting, by introduction of space-filling curves. It is known [Crandall 2000, and references therein] that Hilbert curves, for example, can be generated in linear time (i.e., with operation complexity $O(Db)$ for b-bit coordinate resolution) in any dimension D. This means that we can think of real numbers $t \in [0,1)$ as representing *all* points in $R = [0,1)^D$. In fact, any D-dimensional vector r can be thought of as the result of a Hilbert map h, as in:

$$r = h(t).$$

A complete formalism for discrete Hilbert curves can be developed, in which formalism the notation $h_b^d(t)$ would denote a lattic vector in D dimensions, that is mapped from $t \in [0,1)$ with 2^{bd} possible t and r entities; i.e. the real numbers

$$t \in \{0, 2^{-bd}, 2 \cdot 2^{-bd}, \ldots, (2^{bd} - 1) \cdot 2^{-bd}\}$$

are mapped respectively, under h_b^d, to a total of 2^{bd} vectors in the unit D-cube R.

Now the discrete Hilbert mapping admits of an exact inverse (even though the classical, continuous Hilbert curve does not). It is also the case that the Hilbert map is measure-preserving, in the clear sense that appropriately-justified subregions of the real interval $[0, 1)$ are mapped into "fair" volumes. For example, for $D = 2$ dimensions, each of the real intervals $[0, 1/4)$, $[1/4, 1/2)$, $[1/2, 3/4)$, $[3/4, 1)$ is mapped onto a quad sector, a subsquare of the unit 2-square. This means that we have a guiding, universal principle: *random points in D-space can be modeled as random points in $[0, 1)$ via the D-dimensional Hilbert map.*

Thus the localized dart-throwing problem is brought down to the 1-dimensional case in this sense: if one can solve the problem for the "pullback"

$$h^{-1}(S) \subseteq [0, 1),$$

meaning the set of real values whose image under h is the subregion $S \subseteq [0, 1)^D$, then the forward h map on said real values is the desired dart set. Before moving on, we note that space-filling curves are not absolutely necessary to this analysis. One could also simply take axis projections of a subregion S; e.g. checking in $D = 2$ for hard bounds on x, y axes. Then one could solve the subregion-throwing problem on each axis, and so on. Yet, the space-filling curve approach lends us the luxury of only having to think of one pass, in one dimension.

3. An explicit subregion solution

So let us assume a 1-dimensional subinterval $S \subseteq R = [0, 1)$ be given, and we wish to drop darts (point) onto S in our required, consistent fashion (consistent with points that would be dropped over the full region R). We first give an algorithm that requires as input an overall density N of points (which now is the same as specifying N total random points over all of $R = [0, 1)$):

Algorithm (3.1) for throwing into $S = h([t_0, t_1)) \subseteq R = [0, 1)^D$:

1) Choose overall density N (i.e. N points would drop into the entire interval R), and set $\lambda := 1/N$.

2) Assign integers:

$$k_0 = \lfloor \frac{t_0 - L}{\lambda} \rfloor,$$

$$k_1 = \lfloor \frac{t_1 + L}{\lambda} \rfloor,$$

where L is a characteristic length $>>$ the average separation λ of random points, for example perhaps $L := 10\lambda$;

3) For each $k = k_0, k_0 + 1, k_0 + 2, \ldots, k_1$ calculate the real numbers:

$$T_{k-k_0} := k\lambda + \eta(k, L, \lambda),$$

where η is a random variable drawn from a probability density $f(t)$ that vanishes outside of $t \in [-L, L]$;

4) The desired D-dimensional points to be dropped in the D-dimensional subregion $S = h([t_0, t1))$ are now given by: $\{h(T_0), h(T_1), \ldots\}$.

The workings of the algorithm can be understood in the following way. Imagine the pullback $h^{-1}(S) = [t_0, t1)$ is covered by a slightly larger interval $[t_0 - L, t_1 + L)$. On step (3) of the algorithm we calculate a set of "dithered" points, as perturbed positions of the natural grid points $k\lambda$. So the key to this algorithm is the η random variable, which effectively perturbs the natural grid, and furthermore we only use grid points on and around the desired subinterval. It is clear, by the constraint on the domain of the probability density f of η, that the points dropped into the subregion coincide precisely with the overall dropping restricted to said subregion.

One could argue that the η variable is using recourse; i.e. here is another random function call of unknown seed status. But that is not really so, for example η can be a deterministic hash function applied to its arguments.

As for the statistical quality the algorithm, note that the microscopic probability of a point landing in a tiny interval is:

$$\text{Prob(a point falls in } (t, t + dt)) \sim dt \cdot \sum_{k=k_0}^{k_1} f(k\lambda - t),$$

and this can be made near-uniform in the position t (as it must be, if any subregion $S \subseteq R$ is to receive its fair share of darts) by proper choice of the given parameters, especially the characteristic overlap L.

As for computational complexity, the number of "darts" that must be thrown onto the overlap interval is of order:

$$k_1 - k_0 \sim (t_1 - t_0)N + 2L/\lambda$$

which by the metric property of the space-filling curve is

$$\sim \frac{\mu(S)}{\mu(R)} N + 2L/\lambda.$$

Thus, larger overlap length L gives rise to a more uniform statistic, but adds to the complexity. Still, for fixed λ, L and large N the algorithm obviously reduces the overall complexity by the desired factor of the subregion volume ratio.

4. Solutions via quasi-Monte Carlo

The random variable η in the previous algorithm, with its probability density vanishing outside $[-L, L]$, is not the only way to introduce randomness. Indeed, to allow determinism in the previous solution, η itself must be "advanceable," i.e. deterministic as based on its

arguments alone. There is a different way, however, to generate points in a subregion, and these are not random points of which we speak, but "quasi-Monte Carlo" (qMC) points.

We state right off the primary difference between random and qMC generation: random darts are expected to clump (and gap), whereas qMC points tend to avoid each other, filling in currently open space. When one plots a cloud of random points in $D = 2$ dimensions, say, one sees the expected static with all the fluctuations we intuitively understand. On the other hand, a cloud of qMC points appears smoother, in the sense of a perturbed grid, say. This is why qMC allows for superior numerical integration, by not biasing accidentally any particular region of the domain of an integrand. But it is the perturbed grid idea that underlies the previous algorithm, so we might expect there is a deterministic way to throw qMC darts into subregions.

Indeed, let us ponder a subinterval $[t_0, t1) \subseteq [0, 1)$ and figure out what points of a particular, celebrated qMC sequences—namely the van der Corput 2-adic sequence—fall into said subinterval. The 2-adic generation proceeds elegantly, with the n-th point being the binary number

$$t_n = 0.n_b n_{b-1} \ldots n_1 n_0,$$

where the bits n_i are listed in reverse; i.e., the point-counting index has itself the $(b+1)$-bit binary representation $n = n_b n_{b-1} \cdots n_1 n_0$. Now it is a straightforward number- theoretical issue to determine which reversed-binary integers n fall into a given subinterval.

Similarly, one could avoid space-filling techniques and simply use a D-dimensional qMC sequence, such as a Halton or Sobol sequence, or even one of the newer (t, m, s) net approaches. For $D = 3$ dimensions, one would (on the Halton prescription) use primes $2, 3, 5$ for each of the three dimensions, and block out a subregion $S \subseteq [0, 1)^3$ by bounding the possible values of the respective base-$2, 3, 5$ expansions of indices.

Of course, one could use a qMC approach as the original grid, and perturb said grid according to our previous algorithm. In this way, artifacts due to the exact, natural grid would presumably be dampened or eliminated.

5. Fourier power spectra and "randomness"

It is perhaps easiest to analyze power spectra for algorithms that do not utilize space-fills, rather use simple "jitter" of a justified grid. Note that have previously mentioned that one can avoid space-fill concepts and perform jitter on D independent axes. It is not unexpected that the spectral theory for space-fill generation of darts is extremely difficult. (However, the present author has reason to believe that space-fills will add a desired component of randomness to any scheme—such is in facvt the case in certain qMC research—although this advantage has yet to be quantified in the world of dart-throwing *per se*)

So let us proceed with a simple, 2-dimensional dart-throwing algorithm [Hourvitz 2000], with a view to analysis of its Fourier power spectrum:

Algorithm (5.1) for simple grid jitter ($D = 2$ dimensions)

1) Establish a justified grid and a maximum jump distance L;

2) Jitter each grid point onto a square of apothem L and centered at said grid point;

Note that we have *not* specified the jump distribution, although to be consistent with our promise of a fast localization scheme, we do posit the maximum jump L in each coordinate direction. (One could also contemplate circular symmetry rather than apothem-L symmerty, but the mathematical analysis becomes problematic very quickly).

Armed with this simple algorithm, we proceed to define the resulting dart collection in the form of a delta-function comb distribution:

$$f(r = (x, y)) = \sum_{(u,v)\in\mathcal{G}} \delta^2(r - (u, v) - \Delta_{u,v}),$$

where $\mathcal{G}$ is the justified grid lattice, and the random jitter over a (u, v)-centered square of apothem L is the vector $\Delta_{u,v}$; thus any coordinate displacement satisfies $|\Delta_x| \leq L$ and $|\Delta_y| \leq L$. Next we define a Fourier spectrum by way of the transform defined for 2-vectors a:

$$F(a) = \int e^{ia\cdot r} f(r) d^2 r.$$

This transform has local power $|F(a)|^2$, so that expected statistical power turns out as:

$$< |F(a)|^2 > = \sum_{(u,v)\in\mathcal{G}} \sum_{(u',v')\in\mathcal{G}} e^{ia\cdot((u,v)-(u',v'))} < e^{ia\cdot(\Delta_{u,v}-\Delta_{u',v'})} > .$$

Now all considerations of statistical power can be seen to hinge on the key expectation:

$$E(a, u, v, u', v') = < e^{ia\cdot(\Delta_{u,v}-\Delta_{u',v'})} >$$

which in turn can be—under the assumption of independent (x, y)-jumps, factored into two expectations:

$$E = \left[\int_{-L}^{L} e^{ia_x z} f_2(z, u, u') dz\right] \left[\int_{-L}^{L} e^{ia_y z} f_2(z, v, v') dz\right]$$

$$= \Phi(a_x, u, u')\Phi(a_y, v, v'),$$

where $f_2(\rho)$ is the probability density for the difference of *two* coordinate jumps, for example the difference $\rho = (\Delta_{u,v})_x - (\Delta_{u',v'})_x$; and Φ is the standard characteristic function associated with said probability density. Now the characteristic function Φ is seen to be unity when coordinates (the latter two arguments) coincide. Otherwise Φ depends only on its first argument and can be calculated via the indicated integral. Let us call this characteristic-function eveluation $\epsilon_L(\beta)$ for β a coordinate such as a_x or a_y. In calculating the power $< |F|^2 >$ one has to take into account, as is always the case with noise that fills all of space in some sense, volumetric considerations—after all, we need to normalize because of course the spectral power in any band can be expected to be infinite. Let us therefore take a domain for which $u, v, u', v' \in [0, M - 1]$, that is for some (large) M we restrict the entire procedure to a square of side M. Then the statistical power spectrum works out as:

$$< |F(a)|^2 > = P(a_x)P(a_y)$$

where a power function P is defined thus on one coordinate say β:

$$P(\beta) \propto \epsilon_L(\beta) \frac{\sin^2(\beta M/2)}{\sin^2(\beta/2)} + (1 - \epsilon_L(\beta))M.$$

This power formula is fundamental to the present analysis, in that now we can analyze the amount of power attributable to desired, "white" behavior. In fact, let us write the above power formula in the heuristic form:

$$< |F(a)|^2 >= P(a_x)P(a_y) \propto \ M^2 + (\epsilon_L(a_x) + \epsilon_L(a_y))B(a_x, a_y) + O(\epsilon_L^2)$$

$$= (\text{ "white" } M^2) \quad + \quad (\text{ "bad" terms in } \epsilon_L),$$

where B is a bounded function and by "bad" in this context we mean, of course, correlated noise due to the original, hard lattice positions of the jittered points. The good news is, the "bad" power component is suppressed essentially by factors ϵ_L, which is in turn generally decaying in L. These machinations imply the principle:

Principle: If the characteristic function ϵ_L decays sufficiently fast for large L, then the noise associated with the jittered grid Algorithm (5.1) approaches "white" as $L \to \infty$.

Even though the "bad" component to the spectral noise power has oscillatory components, such oscillation depending in a detailed way on the lattice spacing and the overall domain size M, this bad power is nevertheless damped by the important factors ϵ_L. Let us look at a specific example: if Algorithm (5.1) be used with uniform, square-distributed jitter of apothem L, the damping factor works out to be:

$$\epsilon_L(\beta) = \frac{\sin^2(\beta L)}{\beta^2 L^2},$$

so that:

Principle: For a uniform square jitter of apothem L in Algorithm (5.1), the "bad" (non-white) spectral power component decays as $1/L^2$.

One may take this notion of damping of the "bad" spectral power further. For example, consider the jitter defined by the tent density function:

$$f_1(\beta) = \frac{1}{2L}\left(1 - \frac{|\beta|}{2L}\right)$$

(and f_1 vanishes for $|\beta| > L$) on each axis, so that the joint density for 2-dimensional grid jitter is $f_1(x)f_1(y)$, which is a kind of pyramid with hyperbolic faces. One can work out the characteristic function Φ and the resulting damping function:

$$\epsilon_L(\beta) = \frac{\sin^4(\beta L)}{\beta^4 L^4},$$

on the strength of which we have:

Principle: For a pyramidal jitter of apothem $2L$ in Algorithm (5.1), the "bad" (non-white) spectral power component decays as $1/L^4$.

Naturally, the story does not end here. One may observe that the pyramidal jitter is, in effect, the convolution of two serial jitters, each of the original, uniform-square variety (this is, in fact, why we choose apothem $2L$ for our pyramidal jitter). It certainly makes sense that this double jitter, which amounts to a single-jitter action on a once-jittered grid, would have superior asymptotic properties.

The natural sequence of optimizations would of course end up in a limited-Gaussian jitter, equivalent to say N iterations of a jitter starting with a hard-justified grid. By "limited-Gaussian" we mean that there needs be a hard boundary to the jitter density function, so that the basic idea of subregion determinism be intact. One interesting avenue of optimization would then be, to find the number of jitter iterates N, and the overall apothem NL which made computational sense. What compounds such a study is the need for about N tims the work in generating jitter by such means. For large enough N, one would not create a limited-Gaussian density for the jitter by direct summation of N jumps; rather one would resort to existing fast algorithms for a Gaussian-like density. Whatever the explicit algorithm for L-limited-Gaussian jitter, the fact of a Gaussian's characteristic function being again Gaussian (and we expect this classical symmetry to carry over approximately for L-limited jitter) we arrive at an unguaranteed but intuitive:

Principle: It should be possible to achieve exponential decay in bad spectral power; i.e. $O(e^{-kL^2})$ damping, by choice of limited-Gaussian approximations for the jitter density in Algorithm (5.1).

We repeat that the use of space-fills to reduce the entire D-dimensional analogue down to 1-dimensional dart throwing has not yet been analyzed for spectral properties, yet this author believes there must be a natural advantage to the use of space-fill (extra randomization in some sense, at the cost of the space-fill mapping times of course). Thus, an outstanding open problem is to determine the modification to the spectral power behavior due to the expedient of space-fills.

The author believes there may well be yet a different route to optimal dart-throwing, which is to start with a qMC grid; i.e. not a hard-justified Cartesian grid but a set of qMC points with equivalent prescribed density. Then it is intuitievly suggested that the power spectrum will have very little of the M-dependent aliasing terms, and superior asymptotic white-noise properties.

Thus, a second outstanding research problem is to determine the analytic spectral properties of a qMC-grid jitter variant of Algorithm (5.1), if not a space-fill variant on top of that variant.

Acknowledgments

This interesting dart-throwing dilemma was posed to the author by L. Hourvitz, who indicated by the spirit and tone of his very posing, as he often does, some avenues through which to arrive at solutions.

References

Crandall R 2000,"Scientific applications of space-filling curves," manuscript in prep.
Hourvitz L 2000, private communication.