The Unreasonable Effectiveness of Quasirandom Sequences (extremelearning.com.au)

140 points by panic on Aug 30, 2018 | hide | past | favorite | 49 comments

mturmon on Aug 30, 2018 | next [–]

Anyone who does Monte Carlo simulations would at least be *interested* in the notion of Quasi Monte Carlo, in which points are sampled "more evenly than random."

As noted in OP, this can lead to O(1/n) convergence of averages to underlying target values, rather than the expected O(1/sqrt(n)) convergence ("n" is the number of probes of the underlying system). This is shown, strikingly, in Fig. 2 of the OP.

There is a conference, tutorial papers etc. E.g.: http://mcqmc2018.inria.fr

> extremelearning on Aug 30, 2018 | parent | next [–]
>
> Absolutely! This is actually the original reason I began exploration into quasirandom numbers. I wanted to see if I could replace some of the stochastic process in current machine learning architectures with quasirandom processes.
>
> In principle, the curse of dimensionality kicks in after a dozen or so dimensions, as convergence is O(log(N)^D / N). But in practice I have found that in a surprising number of applications and circumstances, quasirandom sequences can have offer a substantial improvement even when the number of dimensions are in the hundreds or even thousands. (Some authors have suggested this works because the solution space is of low dimensions but embedded in a much higher dimensional space...)
>
> However, to get the full benefit I needed to find a low discrepancy quasirandom sequence that did not suffer the many of the parameter degeneracy problems that many of the conventional ones exhibit for very large D. For me, the new R-sequence nicely solves this parameter selection problem by not having any parameters to optimize!
>
> > jacobolus on Aug 30, 2018 | root | parent | next [–]
> >
> > Can you explain how you figured out that these were the right generalization of the golden ratio? Were you doing a literature search? Just experimenting?
> >
> > BTW, in the past couple months I have directed several people who were interested in spiral phyllotaxis and the like at your page. I think your sequences could make for some interesting art projects.
> >
> > Personally I plan to try them out for dithering, picking well-spaced colors for diagrams, picking points in a region when trying to optimize a map projection, etc.
> >
> > > extremelearning on Aug 30, 2018 | root | parent | next [–]
> > >
> > > Thanks for the kind words and promotion! ;) These findings were a combination of a lot of literature searching, perseverance, decades of mathematical intuition and computer modelling of various hypotheses. I had a problem to solve so I was determined to find a solution. And then combine this with my obsession with simplicity, it meant I didn't just want to tweak a million parameters to get a result.

cosmic_ape on Aug 30, 2018 | parent | prev | next [–]

Just to make sure, the uniform grid on n points will also lead to O(1/n) convergence, right?

I get that the uniform grid is not an open sequence, just want to understand which kind of convergence do you have in mind. For which class of functions?

> extremelearning on Aug 30, 2018 | root | parent | next [–]
>
> For a known fixed value of n, you can get amazing convergence rates. For example, the basic midpoint rule gives O(1/n^2) and Simpson's rule gives O(1/n^4).
>
> Despite this, there are two major use cases for quasirandom sequences: The first is when n is not known upfront and so you can't create a lattice; and the second is in higher dimensions. The reason for this latter situation is because the number of points required to make a lattice in D dimensions scales exponentially in D, whereas the convergence rates for random or quasirandom sampling are (amazingly!) independent on the dimension.
>
> I think that you would really like Owen's paper on this topic. It is very comprehensive and yet extremely readable [1]
>
> 1[ ]http://statweb.stanford.edu/~owen/mc/Ch-quadrature.pdf
>
> > _0ffh on Aug 30, 2018 | root | parent | next [–]
> >
> > Yes, with a fixed n you can essentially do what some fields call DOE [1]. If your sampling process is open-ended things are different.

[1] https://en.wikipedia.org/wiki/Design_of_experiments

cosmic_ape on Aug 30, 2018 | root | parent | prev | next [–]

I certainly agree re:high dimensions. This is because to integrate, you don't need to approximate the function everywhere.

mlthoughts2018 on Aug 30, 2018 | parent | prev | next [–]

I work professionally in large-scale mcmc projects and don’t find quasi mc to be much more than a curiosity in my work, because integration (as opposed to drawing a rich set of posterior samples) is virtually never the goal.

To do anything of use, you have to draw samples and then simulate complex outcomes, very few of which reduce to simple averages over the drawn samples or functions of the drawn samples. And since you need to draw samples, you might as well use those samples for any parts that do happen to require integration, rather than a quasi mc calculation, except in extremely toy-problem situations where the convergence rate matters disproportionately for that smallset of outcomes.

I agree that for cases when you just want to evaluate an integral it could be useful.

I have never encountered a use case when anyone just wanted to calculate an integral, as opposed to also generating posterior uncertainty metrics, posterior test statistics for posterior predictive checking, posterior diagnostics like ordinal statistics among the posterior samples.

I’m sure outside of stats, there must be use cases. Just adding a counterpoint to the idea that quasi mc should always be interesting to practitioners. For a lot of people who work in mcmc methods, quasi mc is just not interesting and generally speaking could never be.

mturmon on Aug 31, 2018 | root | parent | next [–]

I agree with all of this. QMC is more relevant to Monte Carlo, but MCMC is its own thing as far as I know. There is some work on combining the two, but it doesn’t seem mature.

If you don’t mind: what’s the general application domain for your MCMC‘s? For me, it’s inference of (usually) spatial fields for various science data sets, like Earth or solar imagery or atmospheric composition.

mlthoughts2018 on Sept 1, 2018 | root | parent | next [–]

For me it’s posterior sampling to generate meaningful uncertainty metrics in very large scale causal inference for understanding location-based behavioral data, sometimes for ads, sometimes for other modeling problems like image quality, app engagement, etc.

extremelearning on Aug 30, 2018 | prev | next [–]

Author here. OMG! Someone submitted my post here and it's got to the front page!! Happy to try to answer any questions that people have. Enjoy!

nestorD on Aug 30, 2018 | parent | next [–]

Very nice blog (I subscribed to the newsletter). I am looking forward to the "quasirandom stochastic descent" and "extreme learning machines" sections.

leni536 on Aug 30, 2018 | parent | prev | next [–]

I also played with improving the Halton sequence. There is an other generalization of the Golden ratio, namely the metallic ratios. Unfortunately using them performs worse than the Halton sequence.

leni536 on Aug 30, 2018 | prev | next [–]

Very cool! My only gripe of low discrepancy sequences is how they are used in Quasi Monte-Carlo integration:

```
1/n \sum_1^N f(a_n),
```

so each point gets equal weight. One can generalize this that every point gets it's own weight:

\sum_1^N w_{N,n} f(a_n).

I have never seen this generalization elsewhere. I only study Quasi Monte-Carlo as a hobby, maybe I just missed it. Once I had significantly improved discrepancies (close to theoretical minimum) with a modified Van der Corput sequence and linearly diminishing weights. It's all in one dimension though.

My other idea for two dimensions were to use the Hilbert-curve (the infinite limit) to map a one-dimensional sequence to a square area, but AFAIK this was done by others before.

extremelearning on Aug 30, 2018 | parent | next [–]

Yes, using an open (infinite) one dimensional low-discrepancy sequence in conjunction with a space-filling curve such as the Hilbert-curve has been explored. For example by Owen [1]

Niederreiter himself has a beautiful paper that includes a section on weighted low discrepancy sequence [2] where he

also cites several references within it that might be of interest to you.

[1] http://statweb.stanford.edu/~owen/reports/extgrid.pdf [2] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.614...

SeanLuke on Aug 30, 2018 | prev | next [–]

Forgive me if you make this clear in the text (I may have missed it), but you have various pull quotes which say: "The new R2 sequence is the only 2-dimensional low discrepancy quasirandom sequence where..." Are you saying that it's the only *known* sequence? Or the only one *possible?* If the second, do you have formal proofs for those claims?

extremelearning on Aug 30, 2018 | parent | next [–]

I somewhat stumbled on this during my efforts to improve some machine learning architecture I was working on, so unfortunately no formal proofs as yet. This is one reason why I went down the route of technical blog post rather than more academic routes which would no doubt require more rigour and precise language.

Although in many cases I have reason to strongly believe that it is not possible for other sequences to have similar properties, until such proofs are established, I agree that in a technical paper it would be prudent for me to preface most of the quotes with "only known..."

jacobolus on Aug 30, 2018 | root | parent | next [–]

One thing I wondered is if there's a way to tweak this (e.g. by skipping values in one direction) to have better region filling (with respect to Euclidean distance) when the aspect ratio is not square.

Try sliding the top slider at https://beta.observablehq.com/@jrus/plastic-sequence to see what I mean.

extremelearning on Aug 30, 2018 | root | parent | next [–]

Unfortunately, I haven't found a nice solution to this problem yet. For those requiring quasirandom Monte Carlo integration over a say a 2x1 rectangular grid there is always the option of scaling the unit square points by 2 and then rejecting half of the points. Not super elegant and not very efficient in high dimensions, but at least it is unbiased and maintains the low discrepancy characteristics.

DoctorOetker on Aug 30, 2018 | root | parent | next [–]

While your R2 sequence is obviously better than the others for hyper uniform distribution on the sphere, I can't help but notice there are still spiral patterns. (are we looking at a pole of the coordinate transform btw?).

Is this fundamentally the same or similar problem as the rectangular stretch mentioned here? But with a "jacobian" stretch as opposed to a linear stretch?

Could you please read the short section https://en.wikipedia.org/wiki/Orthogonal_matrix#Randomizatio... ?

It is very short, first it defines what is meant with a random orthogonal matrix, then it describes the problem with an incorrect approach, ... and then it describes a recursive procedure to generate a random orthogonal matrix of dimension D using D * (D+1) / 2 *uniformly* distributed numbers:

>Stewart (1980) replaced this with a more efficient idea that Diaconis & Shahshahani (1987) later generalized as the "subgroup algorithm" (in which form it works just as well for permutations and rotations). To generate an (n + 1) × (n + 1) orthogonal matrix, take an n × n one and a uniformly distributed unit vector of dimension n + 1. Construct a Householder reflection from the vector, then apply it to the smaller matrix (embedded in the larger size with a 1 at the bottom right corner).

I predict this will give a better uniform distribution on that sphere, if you share your code for generation and visualization I *might* try myself :)

Also, I applaud you for the extremely clear exposition of the problem, thought process and analogies presented. The one section I found harder to grasp was the one on dithering, as I don't fully understand the process from input picture to output pixel locations and intensities...

extremelearning on Aug 30, 2018 | root | parent | next [–]

Thank you for your kind words! Regarding the mapping of R_2 to the surface of the sphere, I totally agree that although it is better than other options, it is far from optimal. Also, yes, these visualizations are looking from the North pole downwards as I thought this viewpoint highlighted the differnces and discrepancies the best. It was this clearly suboptimal mapping that spurred me to write a prior blog post on improved ways to place N points on a sphere [1].

I think that much of the problem is not with the particular unit square point distribution but rather with the fact that we are using a mapping (lambert) that has a singularity at each pole.

For constant/given n, there is an excellent reference by Saff on mapping points to a sphere. However, if n is not known (ie you need an open sequence of points) I am not aware of any

method that is better than simply/naively mapping a low discrepancy sequence from the unit square to S2.

I think that the most ideal situation will work directly on the surface of the sphere and consist of rotations from one point on the surface to the other. Thus, although this is not my area of expertise I wouldn't be surprised if methods such as what you cite may assist someone in finding a better spherical distribution.

And finally regarding the section on dithering. Thanks for the feedback. I will add some more background explanations to ensure that it is clearer for those who do not directly work in the graphics rendering space.

[1] http://extremelearning.com.au/evenly-distributing-points-on-... [2] http://www.math.vanderbilt.edu/saffeb/texts/262.pdf

DoctorOetker on Aug 30, 2018 | root | parent | next [–]

>I think that much of the problem is not with the particular unit square point distribution but rather with the fact that we are using a mapping (lambert) that has a singularity at each pole.

I agree that the mapping is causing the remaining issue, and that the singularity is *emphasizing* it, but I *think* it is any stretching of a mapping that is causing the spiral clustering towards the equator.

>For constant/given n, there is an excellent reference by Saff on mapping points to a sphere. However, if n is not known (ie you need an open sequence of points) I am not aware of any method that is better than simply/naively mapping a low discrepancy sequence from the unit square to S2.

I understand the advantage of your R2 method regarding open sequences, and the quasi-magical property you can use it to just keep appending points to the collection without any need to "remember" the points already added to position the fresh point. This is clearly a desirable property.

>I think that the most ideal situation will work directly on the surface of the sphere and consist of rotations from one point on the surface to the other. Thus, although this is not my area of expertise I wouldn't be surprised if methods such as what you cite may assist someone in finding a better spherical distribution.

It seems you misunderstood me thinking I was proposing an alternative to an R_2 based method. On re-reading my comment I see I should have been clearer on what I was proposinng:

I proposed using R_2 for a sequency of 6-dimensional tuples (or perhaps just 5-dimensional if the first random number should be 1). Each tuple generated by your R_2 would be used to construct the 3 dimensional random rotation matrix as described. Each rotation matrix then rotates the same reference point (0,0,1). I was proposing an alternative to the lambert mapping, but still using your R_2 sequence, so you keep having an open sequence of points.

I believe you have discovered something very fundamental here, and I can identify with the person who asked for the 1x2 rectangle: perhaps by modifying R_2 for enough simple variations on the problem we can generalize to arbitrary shapes, manifolds or metrics. I am unsure what the best approach is: modify the core logic for generating the next 2-dimensional point in 1x2 rectangle such that it remains hyper-uniform or treat the R_2 sequence as the fundamental primitive, and generate higher dimensional tuples for the uniform hyper-cube and then use math to somehow project it to 2 dimensions.

Thanks for sharing your insights!

extremelearning on Aug 31, 2018 | root | parent | next [–]

1. Stretching I think you and Jacob are on to something when we realise that the R_2 sequence (and presumably R_D) loses some of its properties when it is naively stretched. I think if we solve this problem then we can make some real progress in the sphere problem.

2. Regarding rotations. Yes, I agree. When i wrote 'rotations' i meant it very generally. Any direct transformations from one point on the sphere to another without requiring intermediate mappings such as on a Torus.

3. 6-tuples. This is a really interesting idea, and sorry I missed it the first time. I will definitely explore this idea more.

DoctorOetker on Aug 31, 2018 | root | parent | next [–]

regarding the nr 1: the reason I'd prefer solving the rectangular version

first is because I see 2 potential sources of problem: on one hand an anisotropic but still *flat* metric (i.e. no curvature) and on the other hand the shape or topology of spaces like sphere (which admits elliptic metrics) or double torus (hyperbolic geometry) which force curvature on the metric. That the problem arises already for a flat metric is why I think it deserves attention.

Although the problem can be seen with visual inspection, it incurs some effort to inspect. I propose the following visualization: after generating say N=1000 points, for each point select the closest neighbour, and plot its distance with respect to the origin, then ideally one would get a circular band centered on the origin. Alternatively, plot the angle for each nearest neighbour vector and ideally one should see a uniform distribution.

It's unclear which part(s) need to be generalized for the rectangular version: the equation for the phi's? some rescaling of the phi's? the update rule itself? or some function on the output of the update rule?

Another approach may be to change the concept of hyper uniformity and generalize it to "hyper convergence of [arbitrary function]" i.e. can we have fast convergence to a hat or tent function? If we take the 2-dimensional R_2 and compute the sum of the pair of pseudo-independent numbers, does it generate a tent function? still faster than random sequence? Instead of looking at the distance between adjacent points for the non-uniform tent function, we might multiply this distance by the distance to 0 or 2 (whichever is closest). I.e. points near the floor (0 or 2) should be further apart but will have a lower multiplier since its close to 0 or 2, while points near the top of the tent function (1) will have small distances between them but a higher distance fromm 0 or 2.

For a couple of moments I incorrectly conjectured that spaces have "magic geodesic steps" i.e. for a torus moving straight up or straight to the right does not create a dense point set on the unit square, while the R_2 step does. The reason I think there is no general "magic geodesic step" or at least that it is less related to geodesics as I thought is because *every geodesic* on the unit sphere turns back on itself.

jacobolus on Aug 30, 2018 | root | parent | prev | next [–]

I am also interested in points on the sphere.

I was thinking about using a projection based on mapping a square to an octahedron (along the general pattern of the “quincuncial projection”, but perhaps preserving areas).

However, the topology on the square is not a torus formed by associating opposite sides (as in your pattern) but instead has the two halves of each side of the square folded onto each-other. I suspect the points in the sequence won’t be quite as well distributed near those seams.

jacobolus on Aug 30, 2018 | root | parent | prev | next [–]

> rotations from one point on the surface to the other

This is an interesting idea. You might want to keep track of the previous rotation, rather than just the previous point. You could do some kind of search over the space of unit quaternions to find ones which effectively distributed points.

tlb on Aug 30, 2018 | prev | next [–]

I'm going to try this instead of Halton sequences in my current project.

One detail of the implementation that may be worth improving:

```
z[i] = (seed + alpha*(i+1)) %1
```

When i gets large, say 2^30, you're losing 30 bits of precision in the %1, so (with IEEE doubles) it can only take one of 2^23 values.

The usual implementation of Halton sequences get slower with large values of i, but doesn't suffer from quantization.

extremelearning on Aug 30, 2018 | parent | next [–]

Thanks tlb. You're totally right. I wrote the example code to match the notation used in the post which is a reflection of my maths background. I suspect that from a programming perspective using the recurrence relation:

z[i+1] = (z[i] + alpha) %1

is better practice in terms of both speed and accuracy. I have updated my post to make this clearer. Thanks.

infogulch on Aug 30, 2018 | prev | next [–]

> The methods of creating fully deterministic low discrepancy quasirandom sequences in one dimension are extremely well studied, and generally solved.

Are you referring to something like quadratic residue [1]? It's basically that the sequence x^2 mod P where P is prime appears to be somewhat random, and it goes through the whole sequence before repeating. But it's obviously finite, and it's not perfect [2]. I'm mostly just curious how it relates.

[1]: https://en.wikipedia.org/wiki/Quadratic_residue [2]: https://arxiv.org/ftp/arxiv/papers/1612/1612.05852.pdf

extremelearning on Aug 30, 2018 | parent | next [–]

My intent when I wrote that phrase was two-fold. Firstly to emphasise the large body of work that was done by Weyl, Kronecker, Hurwitz et al relating to the equidistribtuion theorem, badly approximately numbers and diophantine approximation. Armed with this knowledge it is easily proved that the recurrrence relation using the golden ratio mod 1 is optimal using almost any measure.

Although some other readers on HN may be able to see a connection between one dimensional quasirandom sequences and quadratic residue, unfortunately I can’t immediately see an explicit one. Sorry!

My second intent was to contrast this with how little is *provably* known for higher dimensions. To my knowledge, thanks to the phenomenal work by Halton, Sobol, Niederreiter et al, we know that many of the contemporary sequences are optimal in the limiting Big O sense, but there are no proofs even for d=2 for optimality for any of the sequences for general finite values of n.

targafarian on Aug 30, 2018 | prev | next [–]

I'm curious why Sobol outperforms the R-sequence for numerical integration of an 8-dimensional Gaussian and whether that result holds in lower dimensions (and whether that's tied to the infintely-long-tailed nature of the Gaussian; maybe there's a bias in Sobol that is beneficial in this particular case...?).

extremelearning on Aug 30, 2018 | parent | next [–]

Sobol's sequence is really quite amazing. However, there are three reasons why it doesn't get as much attention as it probably deserves.

First is that the maths is far more complex than that of other sequences. Here is a the easiest description I have seen so far [1].

Secondly this complexity as well as the non-trivial requirement to carefully select the basis parameters means that the computing side is very very complex. To get an indication of its complexity, the code for Sobol generation in Mathematica and WolframAlpha is not done by Wolfram itself but rather "comes courtesy of the Intel MKL libraries,... Specifics of the implementation, such as choice of initial values, are not documented. Evaluation of this implementation in terms of discrepancy, projections, and performance in application remains for future work." [also 1]

Thirdly, as defined by d* discrepancy, (which is by far the most common formal method of quatifying discrepancy) the Sobol sequence does not have an asymptotic discrepancy as low as many of other sequences.

Despite this last point, it has been found that in practice Sobol generally yields as good or better performance when used for numerical integrations.

I personally have a suspicion that this is because of the special property, called "Property A" [2] that all Sobol point sequences possess. However, I will let people much smarter than me comment on whether this belief is justified or not...

In one of my other posts [3], Sobol sequences outperform all of the other low discrepancy sequence (including my new R2 sequence) hands down.

Finally, please note that for brevity and clarity, my blog only shows one example for most topics. I will leave it to other authors, or future posts to more rigourously show how consistent the comparisons are....

[1] http://www.mathematica-journal.com/2011/12/a-toolbox-for-qua... [2] http://www.andreasaltelli.eu/file/repository/HD_SobolGenerat... [3] http://extremelearning.com.au/how-to-generate-a-sequence-of-...

Tomte on Aug 30, 2018 | prev | next [–]

[I failed to reply to the author, but I guess you'll see it here, as well]

Please remove the plea for upvotes from your site.

The mods usually react harshly to this and bury the submission.

Your article doesn't need that to stay some time on the front page, anyway. It's high-quality and people have recognized it as such.

extremelearning on Aug 30, 2018 | parent | next [–]

Thanks for this advice. I am new to blogging and certainly new to HN front page etiquette. I think my excitement got

the better of me! I have now taken it off.

Tomte on Aug 30, 2018 | root | parent | next [–]

Great, thanks!

uniqueid on Aug 30, 2018 | prev | next [–]

Neat! Looks like a pretty decent way to generate colored noise. Wish there were an audio demo.

extremelearning on Aug 30, 2018 | parent | next [–]

I have now inserted some sound demonstrations into my blog post! You can now listen to what 1-dimensional (mon) and 2-dimensional (stereo) quasirandom sound might be like.

I will leave it to you to decide which ones you prefer, and maybe even think why some of the quasirandom sequences are more pleasant than others.

For those who have already visited the site, you may need to clear your browser cache to see/hear this update.

uniqueid on Aug 30, 2018 | root | parent | next [–]

Thanks! Interesting. At some point I want to play around with this myself. I'm curious to see what it would sound like to combine the x and y values together (eg: if the frequency is 10 samples between each x, randomly choose between x=0 to x=10 for the first, between x=10 to x=20 for the second). That would probably make it sound less like an FM tone, and more like hissing. I'm not much for math, but a lot of tutorials on colored noise are based on multiple filters. If this algorithm works, it would just need a single low-pass filter for smoothing.

extremelearning on Aug 30, 2018 | parent | prev | next [–]

Only blue noise... I have absolutely no idea what it would sound like. So that means I'll definitely try it later tonight and update the blog post. Tx

uniqueid on Aug 30, 2018 | root | parent | next [–]

Judging by the nicely spaced points in the first animation on your web page, probably pretty good.

bzzzzzt on Aug 30, 2018 | prev | next [–]

This looks similar to the 2D "random number generators" at http://pippin.gimp.org/a_dither/ but is even more evenly distributed.

extremelearning on Aug 30, 2018 | parent | next [–]

Yes. I think that using the dither matrix based on the R_2 sequence should be competitive with any of the the dither masks mentioned on that link.

Thus in situations like video, and/or ray tracing where computing speed is of the essence, maybe a better mask such as the R2 dither mask would offer an improvement on existing methods.

However, in terms of end-quality I am not aware of any dither masks that are anywhere near as good as the state-of-the-art error diffusion algorithms. Thus, for a small number of images such as our favourite animated gifs, I would still be recommending an error-diffusion method. :)

strainer on Aug 30, 2018 | prev | next [–]

I would like to be able to make this sequence, it looks nice. ~~Unfortunately the included python code is not valid~~. (*) It will take me quite a while to grok the math discussion well enough to implement it.

* - pasted it wrong.

extremelearning on Aug 30, 2018 | parent | next [–]

That's strange. To double-check that i typed it into my blog correctly, I copy-and-pasted the code from the post into my python IDE and it compiles and runs properly in my environment. For what it is worth, I am using Python 2.7 on a 64-bit windows machine. Maybe someone else reading has some ideas?

You should notice that after my example code, I have included two other people's code demos so that you can see the same algorithm from a different perspective. Maybe these will help you understand what's required.

In summary, for 2 dimensions, x and y coordinates of the n-th term (n = 1,2,3,....) are defined as:

```
g = 1.32471795724474602596090885447809
a1 = 1.0/g
a2 = 1.0/(g*g)
x[n] = (0.5+a1*n) %1
y[n] = (0.5+a2*n) %1
```

where %1 is the mod 1 operator and takes the fractional part of the argument. Hope that helps.

strainer on Aug 30, 2018 | root | parent | next [–]

I pasted it wrong *blush*

And, Thankyou that seems as clear as can be, when I get my head screwed on straight I will have a better read through your fine insights of the workings.

I will be trying include your sequence in my quirky collection of random helpers for Javascript, illustrated here. http://strainer.github.io/Fdrandom.js/

And will be sure to attribute you and message when I get it in.

extremelearning on Aug 30, 2018 | root | parent | next [–]

That's great. I'm glad you got it working. ;)

PavlikPaja on Aug 30, 2018 | prev [–]

Please note the article only shows correctly in Chrome.

PavlikPaja on Aug 30, 2018 | parent [–]

It seems to be fixed now.