Sixteen Optimal Constants for Multiplicative Hashing

Mark Deaconu

August 2026

This summer I found a connection between two things that have no business being connected: quantum circuit synthesis and hash tables. The bridge between them is arithmetic number theory, specifically the continued fraction of a single real number. This is a note about how that bridge works, and about the sixteen numbers that live on it.

1 Hashing

Multiplicative hashing maps an integer key k to bucket index

$$h(k) = \lfloor B \cdot \{k \cdot \theta\} \rfloor,$$

where $\theta \in (0, 1)$ is a fixed constant, $\{\cdot\}$ denotes fractional part, and B is the number of buckets [1].

Dividing a unit-circumference circle into B equal sectors, labeled in order, with a distinguished origin point at the boundary between sectors $B - 1$ and 0, yields a geometric interpretation of this process. Advancing the origin by k successive clockwise rotations of arc length θ lands on the sector corresponding to $h(k)$, as shown below.

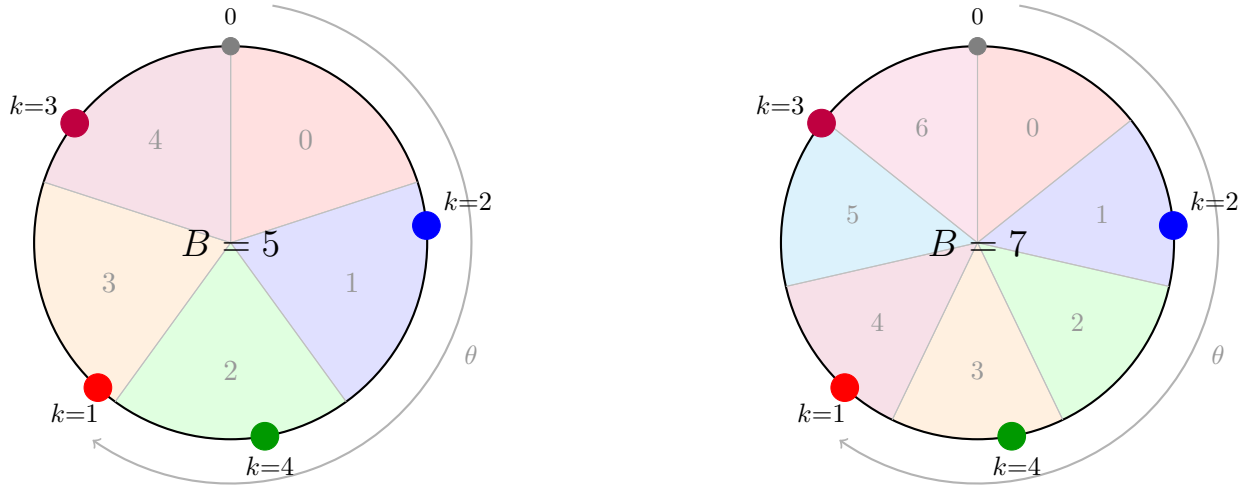

Figure 1: Multiplicative hashing with $\theta = \varphi^{-1} \approx 0.618$. The orbit points are the same in both pictures, only the number of buckets changes.

Multiplicative hashing is the simplest nontrivial hashing algorithm, and appears in the linux kernel primarily for hashing integers (block numbers, IPs, UIDs, GFNs, etc.). At `include/linux/hash.h` we can see that the constant $1 - \varphi^{-1}$ is chosen as the multiplicative constant for both 32 and 64-bit architectures. This constant comes from Knuth [1], who recommends φ^{-1} (and $1 - \varphi^{-1}$ by symmetry) for multiplicative hashing. His argument starts by asking how fast the walk in Figure 1 fills the circle. If θ is close to a rational p/q with small q, the walk is almost periodic with period q, so it keeps revisiting the same q neighborhoods while leaving big arcs empty. The best θ is therefore the hardest number to approximate by rationals. Continued fractions make this precise: the partial quotients of θ control how good its rational approximations are, and the smallest possible partial quotients are all ones. The number with all ones is $\varphi^{-1} = [0; 1, 1, 1, \ldots]$, whose convergents are ratios of consecutive Fibonacci numbers, the slowest growing denominators possible. In this sense the golden ratio is the "most irrational" number, and Knuth's choice is the walk that spreads out the fastest.

2 Generating $U(1)$

On the synthesis side of my work, the question looks completely different. Given a fixed gate set, we want to build arbitrary unitaries out of short gate sequences, and the basic building block is a single rotation. Take one qubit rotation by an angle $2\pi\theta$. Its powers give

$$0,\ \theta,\ 2\theta,\ 3\theta,\ \ldots \pmod 1,$$

an orbit on the circle $U(1) = \mathbb{R}/\mathbb{Z}$. If θ is irrational the orbit is dense, so the gate topologically generates the circle: anything can be approximated if you wait long enough. But dense is not the same as efficient. The practical question is how evenly the first m powers cover the circle, because that is what controls how many gates you need to hit a target.

The natural measure is the largest empty gap $d_\theta(m)$: the longest arc containing none of the first $m+1$ orbit points. Since $m+1$ points make $m+1$ gaps averaging $1/(m+1)$, the ratio $(m+1)\, d_\theta(m)$ is always at least 1, and smaller means a better generator. Graham and van Lint showed that for every irrational θ,

$$\limsup_{m\to\infty} (m+1)\, d_\theta(m) \ \geq\ \rho := 1 + \frac{2}{\sqrt{5}} \approx 1.8944,$$

with equality exactly when the continued fraction of θ is eventually all ones, i.e. θ is related to φ by a $\mathrm{GL}_2(\mathbb{Z})$ transformation. So the same continued fraction that Knuth cared about decides the quality of a generator.

Stier [2] then asked the sharper question: which angles never exceed this bound, at any m, not just in the limit? The answer is a finite set. Exactly 16 values of $\theta \bmod 1$ satisfy $D(\theta) = \sup_m (m+1)\, d_\theta(m) = \rho$. They come in 8 complementary pairs $\{\eta, 1-\eta\}$, with

$$\eta_1 = \tfrac{13+\sqrt{5}}{82}, \quad \eta_2 = \tfrac{7-\sqrt{5}}{22}, \quad \eta_3 = \tfrac{11+\sqrt{5}}{58}, \quad \eta_4 = \tfrac{5-\sqrt{5}}{10},$$

$$\eta_5 = \tfrac{9+\sqrt{5}}{38}, \quad \eta_6 = \tfrac{25-\sqrt{5}}{62}, \quad \eta_7 = \tfrac{3-\sqrt{5}}{2}, \quad \eta_8 = \tfrac{7+\sqrt{5}}{22}.$$

Each one has a continued fraction made of a short prefix (at most 5 terms) followed by all ones. The prefix is what lets the bound hold at every finite horizon, and bounding the prefixes turns the classification into a finite enumeration.

Here is the bridge. The orbit that defines a good topological generator of $U(1)$ is, symbol for symbol, the orbit that multiplicative hashing walks on the circle. Circuit synthesis asks which rotation generates the circle most evenly. Hashing asks which constant spreads consecutive keys

most evenly. It is the same question, asked fifty years apart by two communities that, as far as I can tell, never cited each other.

3 Sixteen constants you can type

Hardware does not store reals, so the constant becomes an odd w-bit integer $a \approx 2^w\theta$, and the hash is

$$h(k) = (a \cdot k \bmod 2^w) \gg (w - \ell), \qquad B = 2^\ell.$$

Quantizing all sixteen (take $a = \lfloor 2^w\theta \rfloor$ and force the low bit) gives constants you can paste directly into code:

η	θ	a_{32}	a_{64}
η_1	0.18581	`0x2F90F67B`	`0x2F90F67B28916D2D`
η_2	0.21654	`0x376F5207`	`0x376F520668CAAEE7`
η_3	0.22821	`0x3A6BD80F`	`0x3A6BD80F395AD825`
η_4	0.27639	`0x46C1B475`	`0x46C1B47480244D95`
η_5	0.29569	`0x4BB213E1`	`0x4BB213E1578AA837`
η_6	0.36716	`0x5DFE35E1`	`0x5DFE35E13DF556D7`
η_7	0.38197	`0x61C88647`	`0x61C8864680B583EB`
η_8	0.41982	`0x6B796829`	`0x6B79682822D839D3`

The other eight are the complements $1 - \eta_i$. Two of the sixteen are already famous. The bold row is exactly `GOLDEN_RATIO_32` and `GOLDEN_RATIO_64` from the linux kernel, so the kernel has been running on η_7 all along. The classic `0x9E3779B9` from TEA is $\eta_7' = \varphi^{-1}$, Knuth's original pick.

One thing that needs checking is whether the circle theorem survives quantization, since the machine orbit lives on $\mathbb{Z}/2^w\mathbb{Z}$ and not on the real circle. The error is controllable: quantization shifts each orbit point by at most 2^{1-w}, so after m steps each gap changes by at most $2m \cdot 2^{1-w}$, and Stier's bound transfers as long as $m^2 \ll 2^{w-2}$. I tested this directly. For consecutive keys, all sixteen constants give max bucket load at most 2 at every table size from 2^8 to 2^{16} on 32-bit words. Out of 100,000 random odd multipliers, the worst had max load 128 at 2^8 and 34 at 2^{16}. So the sixteen come with a guarantee that a random constant simply does not have. The guarantee does degrade where the error term predicts it should: at 2^{20} buckets, two of the sixteen slip to max load 3 and 4. Each hash costs about a nanosecond, so none of this is buying speed, only distribution quality.

4 Where the golden ratio loses

The same arithmetic that makes φ^{-1} optimal also gives it a specific blind spot, and finding it was my favorite part of this whole thing. Hash keys that form an arithmetic progression with step d see the effective rotation $\{d\theta\}$. If $d = F_k$ is a Fibonacci number and $\theta = \varphi^{-1}$, Binet's formula gives

$$F_k \cdot \varphi^{-1} = F_{k-1} + \frac{(-1)^k\, \varphi^{-(k+1)}}{\sqrt{5}},$$

so $\{F_k\varphi^{-1}\} = O(\varphi^{-k})$. The effective multiplier collapses toward zero exponentially fast in k. The Fibonacci numbers are exactly the denominators of the golden ratio's convergents, so stepping by a Fibonacci number resonates with the constant and the walk nearly stands still. Concretely, hashing

4096 keys spaced $F_{11} = 89$ apart into 4096 buckets gives max load 9 under η_7, while every one of the other fifteen stays at max load 2. The most irrational number is uniquely vulnerable to its own convergents.

This generalizes. For any step s, if $\{s\theta\}$ is close to a rational p/q with small q, the walk cycles through about q positions and the keys pile into about q clusters. Kernel pointers are 16-byte aligned, so $s = 16$, and

$$\{16 \cdot \eta_7\} \approx \tfrac{1}{9}, \qquad \{16 \cdot \eta_5\} \approx \tfrac{19}{26}.$$

Under η_7 the aligned pointers see only 9 effective positions; under η_5 they see 26. In a benchmark of 64-bit kernel-style pointers hashed into 2^{16} buckets, η_7 finishes dead last of the sixteen and η_5 beats it by a factor of 4.6 on chi-squared. Page-aligned keys tell a similar story, with η_4 beating η_7 by a factor of 13.

I got excited here and checked whether the kernel should actually swap constants. The honest answer is no. The real call sites of `hash_64` mostly hash plain integers into small tables (64 to 4096 buckets), where all sixteen constants land within a factor of two of each other, and the genuinely hot paths in the kernel do not use `hash_64` at all. The point of this note is the bridge, not a kernel patch.

5 The bridge

The picture I keep coming back to is this. On one side there is circuit synthesis, where you want one gate whose powers fill $U(1)$ as evenly as possible. On the other side there is hashing, where you want one constant whose multiples fill the bucket circle as evenly as possible. Both questions are answered by the same object: the continued fraction of the rotation angle, and the sixteen numbers whose continued fractions are a short prefix followed by all ones.

Stier proved his theorem for reasons internal to number theory and dynamics. Knuth made his recommendation from the algorithmic side. Neither needed to know about the other, but the arithmetic did not care, and the linux kernel has been quietly running one of the sixteen for decades. I like this story because it is the cleanest example I have personally touched of a pattern I want to keep chasing: a very practical engineering object (a magic hex constant in systems code) turning out to be one of exactly sixteen solutions to an abstract optimization problem on the circle. The bridge was always there. You just have to notice that the orbits are the same.

References

[1] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, §6.4. Addison-Wesley, 1973.

[2] Z. Stier, "Optimal topological generators of $U(1)$," arXiv:2002.03092 [math.NT], 2020.

[3] Linux kernel, `include/linux/hash.h`, `GOLDEN_RATIO_32`/`GOLDEN_RATIO_64`.