Injectivity of fraction to float conversion

Mark Dickinson

August 21, 2022

It can occasionally be convenient to store fractions as floating-point numbers. Any mapping from the (infinite) set of all possible fractions to a finite set of limited-precision floats must necessarily be lossy: there will be examples of different fractions which map to the same float. But with a small enough bound on the numerator and denominator of the fractions considered, the function which maps each such bounded fraction to the nearest exactly representable float becomes injective, making it possibly to unambiguously recover the fraction from its floating-point representation. The best possible bound will depend on the target floating-point format.

This note establishes that for IEEE 754 binary64 floating-point (often referred to as "double precision"), 67114657 is such a bound. This bound is best possible, since the distinct fractions 67114658/67114657 and 67114657/67114656 round to the same IEEE 754 binary64 float under round-to-nearest.

Theorem 1. *Suppose that a/b and c/d are rational numbers, written in lowest terms (with b and d positive), that $\max(|a|, b, |c|, d) \le 67114657$, and that a/b and c/d become equal when rounded to the nearest finite IEEE binary64 floating-point number. Then $a/b = c/d$.*

Proof. It's straightforward to check that when we round a fraction with numerator and denominator bounded by 67114657 to a float, no underflow or overflow occurs, and that under the same bounds positive fractions round to positive floats and negative fractions to negative floats. So if a/b and c/d round to the same float then both are positive, both are negative, or both are zero, and in the last case we're done. Negating both fractions if necessary, without loss of generality we can assume going forward that all of a, b, c and d are positive.

Now we separate into two cases. The first case we consider is the case in which a/b and c/d belong to the same closed binade—that is, there's an integer e such that both a/b and c/d lie in the interval $[2^e, 2^{e+1}]$. Since consecutive IEEE 754 binary64 floats in that interval have difference 2^{e-52}, for a/b and c/d to round to the same float we must have $|a/b - c/d| \le 2^{e-52}$, from which

$$2^{52-e}|ad - bc| \le bd. \tag{1}$$

From this inequality combined with $2^e \le a/b$ and $2^e \le c/d$, we have:

$$2^{52+e}|ad - bc| \le 2^{2e}bd \le ac. \tag{2}$$

Using (1) if $e \le 0$ and (2) if $e \ge 0$, along with the bound on $\max(a, b, c, d)$, we have

$$2^{52+|e|}|ad - bc| \le 67114657^2 < 2^{53}. \tag{3}$$

hence

$$2^{|e|}|ad - bc| < 2. \tag{4}$$

It follows that either $|ad - bc| = 0$ (in which case $a/b = c/d$ as required), or $|ad - bc| = 1$ and $e = 0$. In this case the inequality (1) gives $2^{52} \leq bd$, and the inequalities $2^e \leq a/b$ and $2^e \leq c/d$ give $b \leq a$ and $c \leq d$. At this point we look for a contradiction.

Swapping a/b and c/d if necessary, we can assume that $d \leq b$, so from $2^{52} \leq bd$ we have $2^{26} \leq b \leq a$. We can further deduce that $d < b$ (if $d = b$ then $1 = |ad - bc|$ is divisible by b, which is impossible), and that $0 < c < a$.

Now it's possible to do an exhaustive search over all pairs (a, b) of integers satisfying

$$2^{26} \leq b \leq a \leq 67114657.$$

There are exactly 16788115 such pairs (reducing to 10204542 after we discard those with $\gcd(a, b) \neq 1$), making this search computationally very feasible. For each pair (a, b) we can use the extended Euclidean algorithm to efficiently find all possible solutions in integers c and d to $|ad - bc| = 1$ with $0 < d \leq b$ (there are at most two), and check that a/b and c/d do not round to the same float, giving us our contradiction.

There remains the second case, where there is no integer e such that both a/b and c/d lie in $[2^e, 2^{e+1}]$. The only way for this to be possible is if there's a power of two separating a/b and c/d; that is, without loss of generality (swapping a/b and c/d if necessary) there's an e satisfying

$$a/b < 2^e < c/d.$$

From the bounds on a, b, c and d, we must have $-26 \leq e \leq 26$. But now 2^e can be expressed as a fraction with numerator and denominator not exceeding 67114657, and since a/b and c/d round to the same float, 2^e (being squeezed between a/b and c/d) must round to that same float. So we can apply the case 1 proof to a/b and 2^e to deduce that $a/b = 2^e$, and again to 2^e and c/d to deduce that $2^e = c/d$, hence that $a/b = c/d$, as required. □

Listing 1 shows Python code for the exhaustive search; rather than stopping once it reaches $a = 67114657$, it keeps going to print all solutions found to $a/b = c/d$. It produces the first result in under 30 seconds on my laptop.

```
from math import gcd


def find_cd(a: int, b: int) -> list[tuple[int, int]]:
    """
    Given a positive fraction a/b (expressed in lowest terms), find both
    fractions c/d which are simpler than a/b and which satisfy |ad - bc| = 1.
    """
    p, q, r, s = 0, 1, 1, 0
    while b:
        x = a // b
        a, b, p, q, r, s = b, a - x * b, r, s, p + x * r, q + x * s
    return [(p, q), (r - p, s - q)]


for a in range(2**26, 2**27):
    for b in range(2**26, a):
        if gcd(a, b) > 1:
            continue
        for c, d in find_cd(a, b):
            if a / b == c / d:
                print(f"{a}/{b} == {c}/{d}")
```

Listing 1: Exhaustive search