

An Alternative to Floating Point Numbers: Computable Real Numbers

Lucas Rodriguez

University of Dallas

May 8, 2025

I. A Problem With Floating Point Arithmetic

- ▶ Floating point numbers
- ▶ Computable real numbers

II. The Proposed Solution

- ▶ Combining multiple approaches
- ▶ Advantages and disadvantages

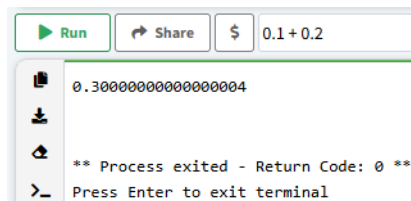
Part I: A Problem With Floating Point Arithmetic

Floating Point Numbers

Floating point numbers (can be) mid. They are good enough, fast, and computationally inexpensive, but can be inaccurate.

Examples:

- ▶ Not associative
 - ▶ $10^{100} + 1 - 10^{100} = 0$ in floating point arithmetic
- ▶ Seemingly wrong results
 - ▶ $0.1 + 0.2 = 0.30000000000000004$



A screenshot of a terminal window. At the top, there is a toolbar with three buttons: a green 'Run' button, a 'Share' button with a circular arrow icon, and a '\$' button. To the right of these buttons is a text input field containing '0.1 + 0.2'. Below the toolbar, the terminal output shows '0.30000000000000004' on the first line. On the second line, it says '** Process exited - Return Code: 0 **'. On the third line, it says '>_ Press Enter to exit terminal'.

Are there any alternatives?

Exact Rational Arithmetic

We can try using exact rationals by representing a number as a pair of 2 unbounded integers.

- ▶ If each of the input numerators and denominators contains n bits, then the output number is represented as roughly $4n$ bits, around as big as the total size of the inputs.
 - ▶ As expressions become deeply nested, this can compound quickly.

Overall, this is not a practical solution to the problem. So what else is there?

Computable Real Numbers

Definition.

Informally, a real number is *computable* if there is an algorithm that prints the decimal digits of the number to any desired accuracy.

- ▶ From this definition, it isn't very difficult to see that rational numbers (and therefore integers since integers are rational) are computable numbers.
 - ▶ Think long division algorithm.
- ▶ So we are more interested in computable irrational numbers.
 - ▶ Familiar irrationals like π and e are computable (think Taylor Series)
- ▶ The existence of computable reals implies the existence of incomputable reals. Luckily for us, most of the numbers we care about for solving problems are computable.

A Quick Note on Approximating Reals With Rationals

Definition.

Informally, a set A is *dense* in B if elements of A are arbitrarily close to elements of B .

- The rationals are dense in the reals.

Definition.

Elements of the set $\{\frac{m}{2^n} \in \mathbb{Q} \mid m, n \in \mathbb{Z}\}$ are called *dyadic rationals*. Equivalently, they are the real numbers that have a terminating binary representation.

- More surprisingly, the dyadic rationals are dense in the rationals.
 - A consequence of this is that real numbers (and therefore computable reals) can be arbitrarily well approximated by dyadic rationals.

Implementing Arithmetic With Computable Reals

Definition.

Informally, a function f is *computable* if there is an algorithm that produces the value of $f(x)$ on input x .

- ▶ We can represent a computable (real) number x in binary by associating it with a computable function $f_x(n)$ that takes in a specified nonnegative integer tolerance n and outputs the nearest unbounded integer m such that the difference between x and $\frac{f_x(n)}{2^n}$ is less than the error tolerance. More explicitly, $|f_x(n) \cdot 2^{-n} - x| < 2^{-n}$.
- ▶ The way in which this is implemented is up to the language being used, but it always can be done for computable reals.

What Can We Do With These?

Without going through the gory details, the computable reals are closed under the basic operations of $+$, $-$, \cdot , $/$.

- ▶ So we can do arithmetic with them to arbitrary precision, which is amazing. No more inaccurate arithmetic (up to desired precision).
- ▶ Additionally, they keep the properties that we want including but not limited to associativity.

The (Practical) Problem With Computable Reals

After establishing these strengths, we must now unfortunately expose the downfall of using computable reals instead of floating point numbers.

- ▶ Put simply, it may take a large amount of memory to actually compute certain numbers to a desired accuracy.
 - ▶ The golden ratio φ , a computable real, is notoriously difficult to approximate by rationals.
- ▶ It is not decidable whether two given computable reals are equal, or if one is greater than the other.
 - ▶ Loosely, something is decidable if there is an algorithm that decides the truth in a finite amount of time.
 - ▶ If you were to type " $1 - 1$ ", the answer is 0, so you want to show "0". But a computer could only tell you " $1 - 1$ is within a rounding error of 0.0000..."

Both of these flaws are massive and undermine the benefit of completely accurate arithmetic.

Part II: A Solution

Restriction of the Computable Reals

We can restrict our attention to the real numbers that can be represented by what are called "calculator operations."

- ▶ The four basic arithmetic operations, and square roots
- ▶ The sin, cos, and tan trigonometric functions and their inverses.
- ▶ Exponential and natural log functions

These are the kinds of numbers we care about for scientific computing anyways, so this is a reasonable restriction, even if we can't reach every computable real this way.

Decidability Resolved

Remarkably, it is decidable whether a constant described by a system of equations over the complex numbers is equal to zero unless the problem contains a counterexample to Schanuel's conjecture (highly unlikely).

The equations can take the following two forms:

- ▶ A (multivariate) polynomial is equal to zero
- ▶ $e^x = y$

Fortunately for us, all of the "calculator operations" can be represented this way.

- ▶ Example: $\sin(z) = \frac{e^{iz} - e^{-iz}}{2i}$

A Compromise Representation For a Real Number

So the solution is the dumb one: just combine all of the approaches.

- ▶ A real number (replacing a float) can be defined with 3 fields:
 - ▶ A rational number (represented by the standard two BigIntegers)
 - ▶ A computable real number (dyadic rational approximation)
 - ▶ A property (essentially storing if we have a special number like π)

The represented real number is the product of the rational factor and the computable real factor.

- ▶ Example: numbers like $\frac{\sqrt{2}}{2}$ are represented exactly as $\frac{1}{2} \cdot \sqrt{2}$.

More on the Property Field

The property field has 2 types of values, a tag with a rational x , or a simple irrationality tag. Essentially, it just tells you whether you have a recognized form of a real number.

► To name a few, we have π , \sqrt{x} , e^x , $\ln(x)$, $\sin(\pi x)$, $\tan(\pi x)$

For example, when adding two numbers whose computable real factors are the same, you just add the rational factors and preserve the computable real factor, just as you expect in infinite precision arithmetic.

► Example: $(1 + \sqrt{2}) + (\frac{1}{2} + \sqrt{2}) = (\frac{3}{2} + \sqrt{2})$

If you add two numbers of different types like $\ln(2)$ and $\sqrt{137}$, then we don't have a special tag like above, but just that the number is irrational. In this case, the stored number gets the irrationality tag.

Strengths

This system of finite-precision arithmetic strikes a decent balance between accuracy and speed.

- ▶ Like the system of computable real arithmetic, the expected properties of infinite precision arithmetic hold.
- ▶ As mentioned before, we recover some decidability of equality this way.
 - ▶ The proof is left as an exercise to the audience.
- ▶ There is some symbolic manipulation for common values with calculator operations, which leads to more exact answers.
- ▶ A nicer user experience when displaying results (less $3.0400\dots$)

Weaknesses

- ▶ Even with the combined approach, this system is still noticeably slower than floating point arithmetic.
 - ▶ Performance and convergence are a large component of numerical analysis, so if speed is important, this system might not be what you need.
- ▶ Not all of the numbers we care about can be represented exactly in this system.
 - ▶ More complicated formulas such as $\pi^2 - \pi^2 = 0$ will not be displayed so nicely.

Some Thoughts

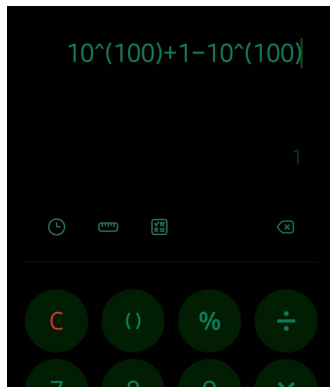
This system of real numbers is not a complete replacement for floating point numbers. It has some advantages, but it makes more sense to use different systems for different contexts.

- ▶ For example, this real number system can be used when there is less calculation and greater accuracy is desired.
- ▶ However, if you need a massive amount of flops, floating point numbers will probably still be better.

An Application

Finally, I would like to end with a cool application of this system to show isn't just the ramblings of a pure mathematician who hates applications.

- ▶ Google implemented this system to the android calculator app to get more accurate answers.
 - ▶ Given that the point of a calculator is to be as accurate as possible, this is the perfect place for such an implementation.



References I

- ▶ Paper I used
 - ▶ <https://dl.acm.org/doi/pdf/10.1145/3385412.3386037>
- ▶ Summary of paper
 - ▶ <https://chadnauseam.com/coding/random/calculator-app>
- ▶ Decidability of solving calculator operation equations
 - ▶ <https://dl.acm.org/doi/pdf/10.1145/190347.190429>

Thank you!