Complex Number Bases

Benjamin Chung

We're all familiar with base 10, the standard number base today. The reason we use it is because humans have 10 fingers. In the past, different civilizations have used different bases: for example, the Mayans used base 20, and the Babylonians used base 60. There is an argument for the use of base 12, because 12 has more divisors than 10, so the arithmetic will have nicer properties. We will probably stick with base 10; if America can't even switch to metric, then the world probably can't switch to another base.

What does it mean to use base 10? It simply means that every digit represents the coefficient of a different power of 10. For example, in base 10,

$$3141.592_{10} = 3 \cdot 10^3 + 1 \cdot 10^2 + 4 \cdot 10^1 + 1 \cdot 10^0 + 5 \cdot 10^{-1} + 9 \cdot 10^{-2} + 2 \cdot 10^{-3},$$

where we use the subscript on the number to represent its base. Typically we omit the subscript 10, and if a number has no subscript it is assumed to be in base 10. In base 5,

$$\begin{aligned} 3141.432_5 &= 3 \cdot 5^3 + 1 \cdot 5^2 + 4 \cdot 5^1 + 1 \cdot 5^0 + 4 \cdot 5^{-1} + 3 \cdot 5^{-2} + 2 \cdot 5^{-3} \\ &= 421.936. \end{aligned}$$

In general, in base B, we use the digits $0,\ 1,\ 2,\ \ldots,\ B-1$ to represent numbers. So, the general formula for a number written in base B is

$$(\overline{d_n d_{n-1} d_{n-2} \ldots d_2 d_1 d_0 d_{-1} d_{-2} \ldots})_B = \sum_{k=-\infty}^{n} d_k B^k$$

where the digits d_k are from the set $\{0,\ 1,\ \ldots,\ B-1\}$. Using this set of digits guarantees two important characteristics of a number system: first, every real number is *representable*, and second, *almost* every real number has a *unique representation*. Why "almost"? It turns out that some numbers can be represented in multiple ways. For example, in base 10, the classic example is that $1 = 0.\overline{9}$. Luckily, it can be shown that a string of repeated nines is the only way to create multiple representations for a base-10 number [1]. When we say *almost* all numbers can be represented uniquely, we mean all but a negligible amount.

That's the reason why irrational bases aren't practical: they allow for a nontrivial amount of multiple representations. For example,

$$4 = (10.22012202112111030100...)_{\pi}$$
$$= (10.21320300121030000210\ldots)_{\pi}$$
$$= (10.21313311212022122230\ldots)_{\pi}$$

[2]. The same problem occurs with fractional bases. Positive integer bases are thus the most practical, because they don't have any of these issues. But what about negative integer bases?

Negabinary and Other Negative Bases

It turns out that the concept of numerical bases extends to the negative integers, as well. As an example,

$$1729.128_{-10} = 1 \cdot (-10)^3 + 7 \cdot (-10)^2 + 2 \cdot (-10)^1 + 9 \cdot (-10)^0 + 1 \cdot (-10)^{-1} + 2 \cdot (-10)^{-2} + 8 \cdot (-10)^{-3}$$
$$= -311.088.$$

Additionally, using the digits $0,\ 1,\ 2,\ ...,\ |B| - 1$ with base B, every real number can be represented, and almost every real number is represented uniquely [3]. This means that negative bases are viable for practical use. But is there any practical value in using a negative base? A potentially useful property is that negative bases can represent both positive and negative values, without using a minus sign. That's one limitation of positive bases: they can technically only represent half the real numbers, because a minus symbol is required for the negatives. This means that negative bases can represent twice the amount of numbers that positive bases can.

Of particular interest is base -2, known as "negabinary." Since it uses the digits 0 and 1, it can also be used by computers, like normal binary. Negabinary actually saw implementation in several computers, such as the 1950s Polish computer BINEG [4], but today we mostly use normal binary. Even though negabinary may not require a minus sign, it often contains more digits than the binary representation of the same number, so it doesn't really save space. Nonetheless, negabinary is a fascinating concept which demonstrates that bases other than positive integers can function just as well. But we can go even further by using *complex* numbers as bases.

Complex Bases

Certain complex numbers can actually serve as number bases. We will look at bases that use Gaussian integers, complex numbers of the form $a + bi$, where a and b are integers. The first question that arises is what digits we will use. For a real integer base B, we would use the digits $0,\ 1,\ 2,\ \ldots,\ |B| - 1$. Remove any one digit, and some numbers are not representable; add another, and a nontrivial amount of numbers could be represented in different ways. These digits form what's called a complete residue system modulo B, which means they represent all possible remainders when dividing an integer by B.

Analogously, a Gaussian integer base $z = a + bi$ must use digits that form a complete residue system modulo z. This is slightly different from a residue system modulo a real number, but Gauss showed that if a and b are relatively prime, then the digits $0,\ 1,\ 2,\ \ldots,\ a^2 + b^2 - 1$ form a complete residue system mod z [5]. In other words, we must use the digits $0,\ 1,\ 2,\ \ldots,\ a^2 + b^2 - 1$ for our complex base.

As an example, let's do a conversion from base 2-i. We must use the digits from 0 to $0,\ 1,\ 2,\ \ldots,\ a^2 + b^2 - 1$.

$$\begin{aligned} 31.41_{2-i} &= 3 \cdot (2-i)^1 + 1 \cdot (2-i)^0 + 4 \cdot (2-i)^{-1} + 1 \cdot (2-i)^{-2} \\ &= 8.72 - 2.04i. \end{aligned}$$

Base -1+i and 1-i

There are two complex bases we will focus on, namely base -1+i and base 1-i, both using the digits 0 and 1. These bases were proposed by Khmelnik in 1964 [6] and Penney in 1965 [7], and they serve as complex analogues to binary and negabinary in the sense that they also use the digits 0 and 1 and thus can be implemented into computers.

Let's first look at base 1-i. To visualize it, we will graph every complex number that it can represent. For now, let's just graph the Gaussian integers for simplicity, and we'll use boxes instead of points to represent the complex numbers. Let's start by graphing all 1-digit base 1-i Gaussian integers (really only 0 and 1):

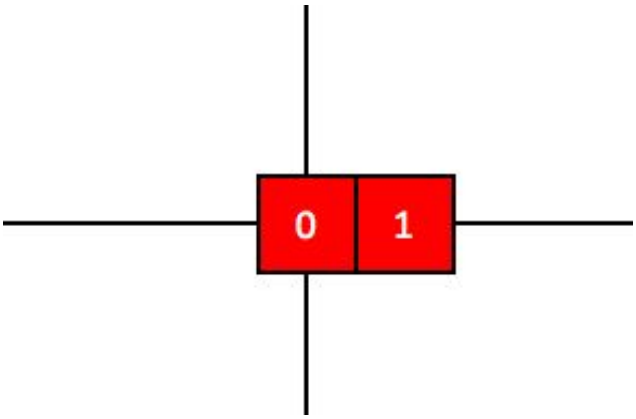

Figure 1: All 1-digit base 1-i Gaussian integers. The numbers in the squares are in base 1-i. Powers of 1-i are in white font.

Not very exciting. Next, we want to graph all 2-digit base 1-i Gaussian integers, and there's a clever way to do this. To get every 2-digit number, we can just add 10_{1-i} to every 1-digit number. Watch the effect on the graph:

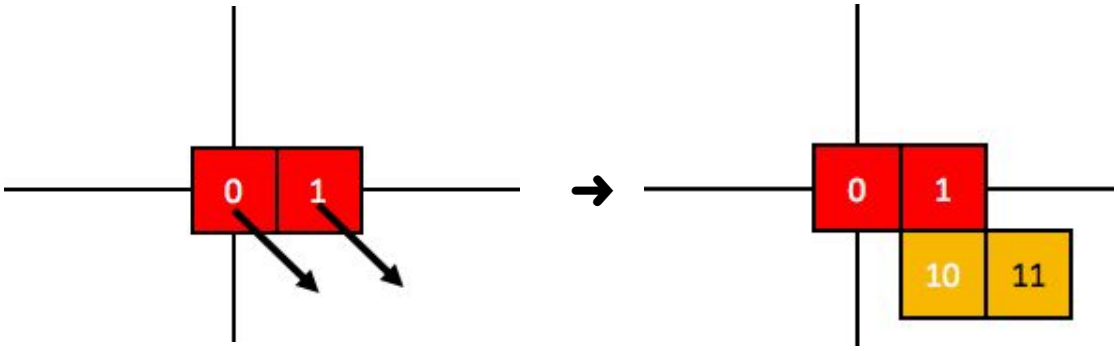

Figure 2: Forming the 2-digit numbers by adding $10_{1-i} = 1 - i$ to each one-digit number. The arrows show how the addition is essentially a translation.

Then, to graph every 3-digit number, we just add $100_{1-i} = -2i$ to every 1- and 2-digit number we already have:

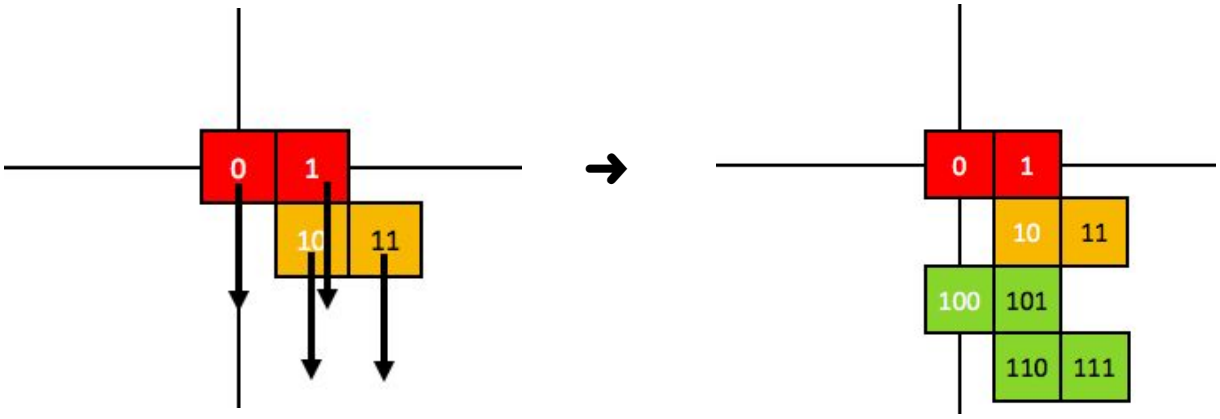

Figure 3: Forming the three-digit numbers by adding $100_{1-i} = -2i$ *to the 1- and 2-digit numbers.*

We can continue this process indefinitely to represent every base 1-i Gaussian integer. In the end, it looks like this:

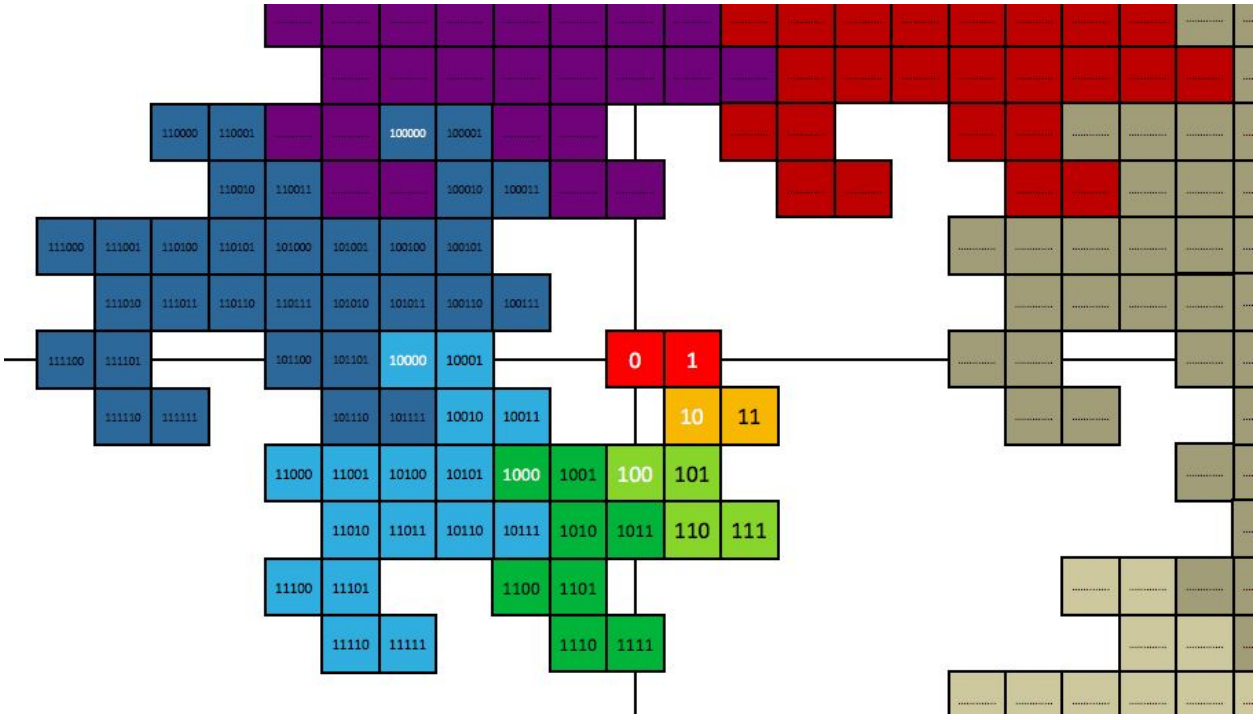

Figure 4: The Gaussian integers representable in base 1-i.

The spiral shape is due to the fact that the powers of 1-i, the ones with white font, spiral outwards from the origin. Each colored region contains numbers with the same amount of digits, and they are all self-similar because of the way we constructed it recursively. On a side note,

these colored regions approach the shape of a fractal called the twin dragon curve (also known as the Davis-Knuth curve), which consists of two dragon curves put together. The dragon curve is a fractal which appears from folding a piece of paper over and over then unfolding it.

Even more remarkable is what happens when we look at *all* complex numbers representable in base 1-i, not just the Gaussian integers.

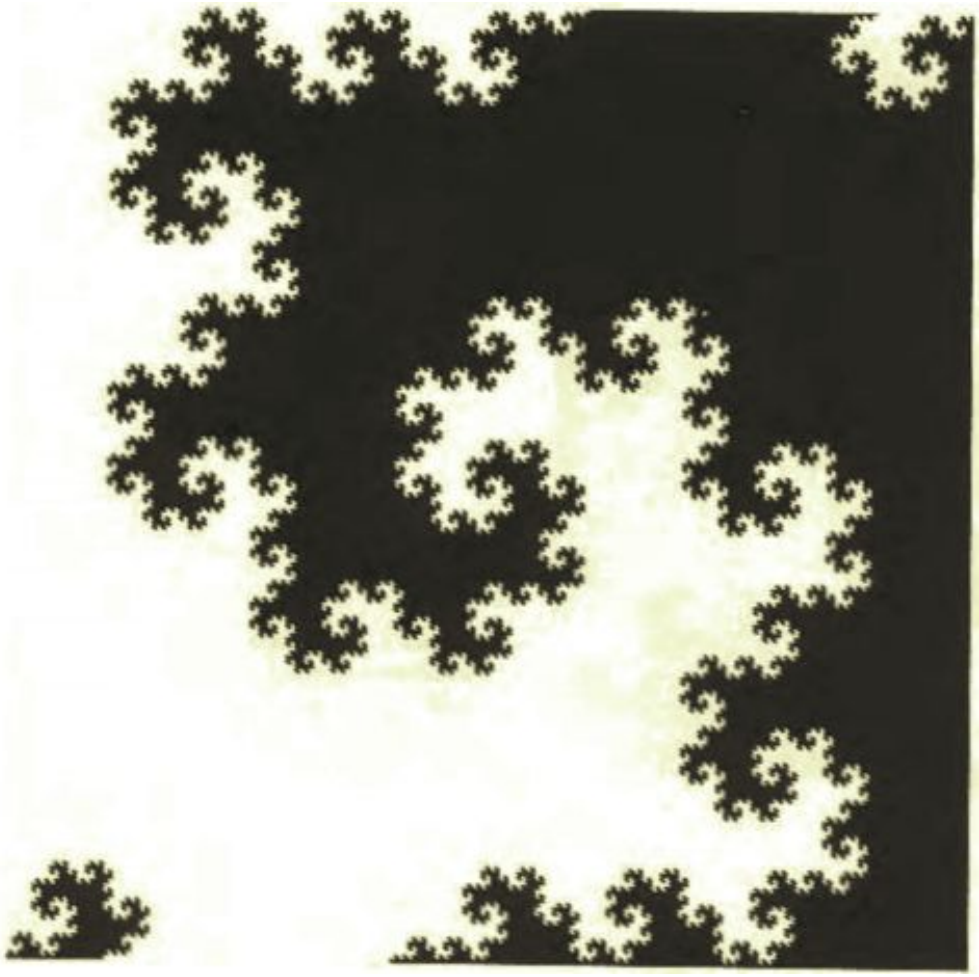

Figure 5: All complex numbers representable in base 1-i (Source: [5]).

This is a very nice fractal. The most remarkable thing is that the white space is exactly the same shape as the black space, which reveals that exactly half of the complex numbers can be represented by base 1-i. In fact, the white space is the reflection of the black space across the line y=-x, which means that each point in the white space is the negative of the corresponding point in the black space. In this way, base 1-i is comparable to binary: they both use the digits 0 and 1, and they both represent only half of the numbers in their systems, as a minus sign is required to access the other half.

Base 1-i is thus a viable option for implementation into a computer, as it can represent every complex number, provided minus signs are allowed. But as binary has a counterpart, negabinary, which eliminates the need for a minus sign, does base 1-i likewise have a counterpart that can represent every complex number without the need for a minus sign? The answer is yes, and it is base -1+i.

Like with base 1-i, let's plot the Gaussian integers representable in base -1+i.

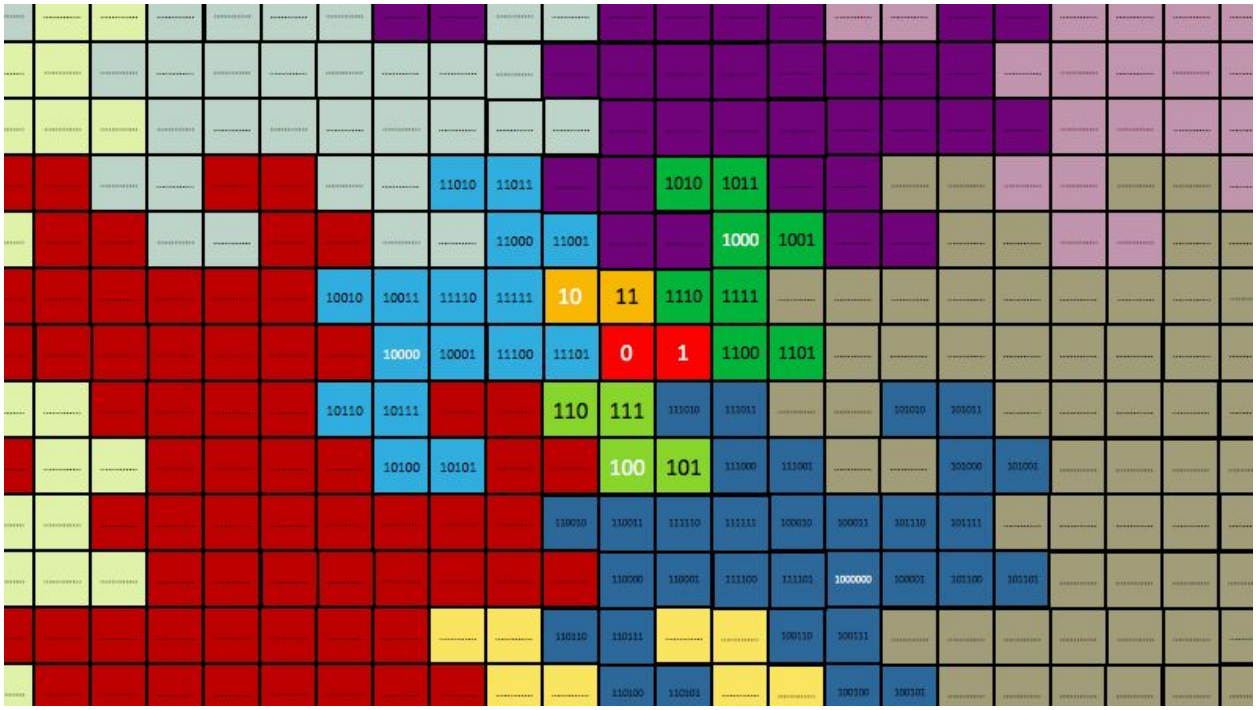

Figure 6: Gaussian integers representable in base -1+i.

The pieces fit together this time, and they end up tiling the whole plane. In fact, the plane is fully tiled even when considering every complex number, not just the Gaussian integers [5]. This means that base -1+i can represent every complex number without using a minus sign, so it is the base with the widest range we've looked at so far.

The connection between base 2, -2, 1-i, and 1+i is quite fascinating. To summarize, base 2 and base 1-i can both only represent half the numbers in their systems, without a sign. Base -2 and base 1+i can represent all the numbers in their systems without a sign. All four bases use only the digits 0 and 1. It is conceivable that there might be other bases analogous to binary and negabinary. Perhaps, for example, there is a quaternion base that can represent all quaternions with the digits 0 and 1, or an octonion base, or a dual number base.

There are many other complex bases that have been studied. One of the most famous is the quater-imaginary base, proposed by Knuth in 1960 [8], which is base 2i with digits 0, 1, 2, and 3. We can look beyond Gaussian integer bases to others like base $\pm\sqrt{2}i$. But base -1+i stands out in the fact that it only uses the digits 0 and 1, so it could potentially be implemented

into a computer. After all, complex numbers have countless applications in computer algorithms, and along with quaternions they are often used in computer graphics. Implementation of complex bases is an interesting idea, and time will tell if it ever works out.

References

[1] Alfeld, Peter. “Uniqueness of the decimal representation of real numbers”. http://www.math.utah.edu/~alfeld/math/sets/unique.html
[2] Kaseorg, Anders. “Can we have a number system with the base pi?”. https://www.quora.com/Can-we-have-a-number-system-with-the-base-pi
[3] https://brilliant.org/wiki/negative-integer-number-base/
[4] Knuth, D. E., 1997: The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 205.
[5] Gilbert, W. J., 1982: Fractal Geometry Derived From Complex Bases, The Mathematical Intelligencer, **4**, 78-83.https://www.math.uwaterloo.ca/~wgilbert/Research/MathIntel.pdf
[6] Khmelnik, S.I. (1964). "Specialized digital computer for operations with complex numbers". Questions of Radio Electronics (in Russian). XII (2).
[7] W. Penney, A "binary" system for complex numbers, JACM 12 (1965) 247-248.
[8] Donald Knuth (April 1960). "An imaginary number system". Communications of the ACM. 3 (4): 245. doi:10.1145/367177.367233.

Special thanks to Wendy Cao.