Chapter 1: Numbers

Section 6 Part 1: Accuracy

We saw that every real number is a limit of rational numbers, and therefore a limit of decimals. So \( \pi \) could be represented as the limit of \((4, 3.3, 5, 2.9, 2.95, 3.28, \ldots)\). We'd rather have it simpler, for example, \((3, 3.1, 3.14, \ldots)\). That is, we'd like to start by picking out the 3 in \( \pi \), then get the .1, and so on.

The closest example of something like this was back on p29: we showed all natural numbers could be written with decimals. This followed from the Archimedean Principle:

**The Archimedean Principle:**
If \( x \) is a positive integer, then there is always a power of ten, \( 10^m \), with \( m \geq 0 \) and \( 10^m \leq x < 10^{m+1} \)

We showed a similar principle for rationals. Because Cantor showed each real is limit of rationals, we can prove:

**The Archimedean Principle for Reals:** If \( x > 1 \) is a real number, then there's an \( m \geq 0 \), with \( 10^m \leq x < 10^{m+1} \).

What we'll try to do next is use this to find the 'decimal expansion' for any real number. We'll make one up: \( r = 632.0141596\ldots \). With \( m = 2 \), \( 100 \leq 632.141596\ldots < 1000 \). Then \( 1 \leq \frac{632.0141596\ldots}{100} < 10 \), and therefore \( r/100 \) has a leading decimal digit (that is, one of the numbers \{0, 1, 2, \ldots, 9\}), plus a fractional part less than 1.

Now subtract the leading decimal digit (6 · 100) from \( r \), and repeat the above, eliminating the 3 and the 2, until you're left with just the fraction (.0141596\ldots) We can also prove an Archimedean principle for fractions:

For positive fractions \( y \) less than one, \( \frac{1}{y} > 1 \), so \( 10^m \leq \frac{1}{y} < 10^{m+1} \), and flipping the inequalities: there is always a power of ten, \( 10^m \), with \( m \geq 0 \) and \( \frac{1}{10^{m+1}} < y \leq \frac{1}{10^m} \).

In our case, \( m = 2 \) and \(.01 < .0141596\ldots \leq .1 \). Again, divide by .01: then \( 1 < 1.4159\ldots \leq 10 \) and once again we have a leading decimal digit plus a fraction. Subtract the 1, and apply the same procedure to .41596\ldots.
What we’ve done is to generate a sequence, \{600, 630, 632, 632.01, 632.014, \ldots\}.

After the second term, we get:

\[
0 \leq (632.0141596 \ldots) - (632.0) = .0141596 \ldots
\]
\[
0 \leq (632.0141596 \ldots) - (632.01) = .0041596 \ldots
\]
\[
0 \leq (632.0141596 \ldots) - (632.014) = .0001596 \ldots
\]

These inequalities are enough to show that 632.0141596 \ldots is the limit of \{600, 630, 632, 632.01, 632.014, \ldots\): all real numbers are limits of very simple finite decimals. We can call this the decimal expansion of real numbers (understanding of course, that .1 and .099999 \ldots are different expansions of the same number).

We now want to move on to a different topic: many (most) real numbers are not finite decimals, and to use them, we have to try to understand them in finite terms. This brings us back to the problem of \frac{1}{7} on p43. We talked about approximations and errors: \frac{1}{7} isn’t a finite decimal, but a number like 0.14285875 is a good approximation. Good because know the error: \text{error} = \frac{1}{7} - 0.14285875. On p43, we saw \(-0.00000235 < \text{error} < 0.00000235\), or, \(|\text{error}| < 2.35 \times 10^{-6}\)

We’ll re-interpret the list of inequalities at the top of this page as errors. We’ll write the real number \(r = 632.0141596 \ldots\), and use \(a_1 = 632, a_2 = 632.01, a_3 = 632.014, a_4 = 632.0141, a_5 = 632.01415\), as approximations by finite decimals.

Then

\[
0 \leq |r - a_1| < .0141596 \ldots
\]
\[
0 \leq |r - a_2| < .0041596 \ldots
\]
\[
0 \leq |r - a_3| < .0001596 \ldots
\]

We wrote these inequalities because they give a special kind of error estimate: they tell how many decimal places of accuracy we have. In particular, \(a_1\) has got the first place to the right of the decimal correctly, \(a_2\) has the second, \(a_3\) the third. So we get more and more accurate decimal places. More than that: once we get, say, the second decimal place right, it stays right.

This is a new way of thinking about error: we could say that \(\pi\) and its approximation 3.145 agree to three decimal places, and this would tell us what finite decimals we could trust. But this, in use, doesn’t work very well. Here’s why.

Let’s take the exact number \(r = 1\), and compare it to approximations \(a = .999\) and \(b = 1.001\). Since \(r = 1.000 \ldots\), \(r\) and \(b\) agree to two decimal places, while \(r\) and \(a\) agree to no decimal places at all.

This suggests we want to rethink ‘number of decimal places’, and
think about errors instead. The error in approximating \( r \) by \( b \) is
\[
| r - b | = .001, \text{ and the error for approximating } r \text{ by } a \text{ is } \\
| r - a | = .001.
\]
So ‘number of decimal places’ and ‘error’ give different standards for accuracy. It gets even trickier: this time take numbers \( a = 3.1451 \) and \( b = 3.1458 \). Each agrees with \( \pi \) to three decimal places. But if we rounded to three decimal places, we’d be ignoring the fact that \( b \) is much closer to 3.146 than to 3.145. So, approximations to three decimal places are not rounding properly.

What we want is an idea that brings together all three ideas: error, decimal places, and rounding. The intuitive idea is to say that \( r \) and \( a \) agree to \( n \) decimal places if they become equal when you round correctly to \( n \) places. Since the cut-off for rounding up or down is 5, we get:

**Definition: Number of Decimal Places of Accuracy**
Numbers \( a \) and \( b \) agree to \( n \) decimal places if
\[
| a - b | < .5 \times 10^{-n}
\]

Repeating the approximations on the previous page,
\[
0 \leq r - a_1 < .0141596 \ldots < .05 = .5 \times 10^{-1} \\
0 \leq r - a_2 < .0041596 \ldots < .005 = .5 \times 10^{-2} \\
0 \leq r - a_3 < .0001596 \ldots < .0005 = .5 \times 10^{-3}
\]

These work out just the way we want: \( r \) and \( a_1 \) agree to one decimal place, \( r \) and \( a_2 \) to two decimal places. And for the example where \( r = 1.000 \ldots, a = .999, b = 1.001 \), both \( | r - b | = .001 \), and \( | r - a | = .001 \). As \( .001 \).\( < .005 = .5 \times 10^{-2}, r, a, b \) each agrees with \( r \) to two decimal places.

One important note: the idea ‘round to the same number’ is only an intuition. If we were to take \( a = .1234 \) and \( b = .1236 \), then we would have \( | a - b | = .0002 < .5 \times 10^{-2} \), so they agree to three decimal places. If you round \( a \) to three decimal places, you get .123, but \( b \) rounds to .124: not the same. The point to this example is that ‘round to the same number’ is the intuition, but the real meaning is given by the definition. It’s what we’ve discussed before: these definitions set standards that everyone can implement in the same way; intuitions don’t have this universality – that everyone gets the same answer.

So, we finally have a standard way to find sequence of rational numbers approximating any real number.
Decimals work well with geometric and analytic intuitions, but can be inconvenient for modern science. Two examples:

\[ \text{i) The absorption of light by a photosynthetic molecule occurs in } \ \frac{0.0000000000015}{\text{seconds}}. \text{ This is also the amount of time that electronic stock market transactions take place.} \]

\[ \text{ii) The amount of CO}_2 \text{ in the atmosphere is increasing, but temperatures on earth haven’t increased as much. A new study shows that from 1865 to 1997, the worlds oceans have absorbed about } \ \frac{150000000000000000000000}{\text{Joules}} \text{ of energy: see Figure 58. (A Joule is about the amount of energy released by dropping a tomato from three feet).} \]

Scientific notation helps deal with zeroes. It counts the number of decimal places to the right of the leading decimal, and puts that into a power of ten. It also requires a single (non-zero) digit to the left of the decimal point. Thus, \( 100 = 1 \times \times 10^2 \); \( 21060 = 2.106 \times 10^4 \); \( 150000000000000000000000 \text{ Joules} = 1.5 \times 10^{23} \text{ Joules}. \)

For a fractional numbers like \( .0231 \), scientific notation counts the number of decimal places to the right of the non-zero leading decimal, so \( .0231 = 2.31 \times 10^{-2} \text{ and } .000000000000015 \text{ seconds} = 1.5 \times 10^{-16} \text{ seconds}. \text{ Thinking slightly differently, } .0231 = \frac{2.31}{10^2} = 2.31 \times 10^{-2}. \text{ In either case, the effect is still to have only a single (non-zero) digit to the left of the decimal point.} \)

Powers of ten are reference points for daily life, but powers of a thousand are more often used in science, engineering, and technology: we have a meter \((10^0 \text{ meters})\), a millimeter \((10^{-3} \text{ meters})\), a micrometer \((10^{-6} \text{ meters})\), a nanometer \((10^{-9} \text{ meters})\), etc. \( 10^{-15} \text{ units} \) is a femto-unit, so the photosynthetic reaction above takes place in 15 femtoseconds. Similarly, \( 10^{21} \text{ units} \) is a zetta-unit, so the amount of heat energy released into the atmosphere is 150 zettajoules.

A system using powers of \( 10^3 \) or \( 10^{-3} \) is called engineering notation. In this notation, \( 21060 \) would be written as \( 21.060 \times 10^3 \), while \( .0231 \) would be \( 23.1 \times 10^{-3} \).

This notation (engineering) is often used to describe modern computers. Consumers look at processor speed: ‘wow that’s a 4 ghz processor’. Scientists prefer to measure how many computations per second a computer can do. Additions or multiplications of natural numbers are called integer operations; those can be very fast. But most scientific computations are with decimals; these are called floating point operations. Scientists and engineers need to know how many floating point operations a computer can perform per second; this is referred to as FLOPS (the FLOP measurement is not easy to calcu-
late; it’s often a measure of an ideal, rather than what a computer can consistently do on real computations; see p73).

To look at some examples of computer speeds, we need terminology. A mega- is $10^6$ (a million); a giga- is $10^9$ (a billion); a tera- is $10^{12}$ (a trillion); a peta- is $10^{15}$ (a quadrillion).

The first commercial supercomputer was the CRAY-1 (Figure 59) introduced to Los Alamos in 1976; it computed at 100 megaflops. In comparison, the first Macintosh from 1984 was about 10,000 times slower. Apple’s fastest computer in 2016, the MacPro, runs at a maximum of 436 gigaflops.

And the 2016 world speed champion was the Chinese Sunway Taihu Light, (Figure 60) running at 93 petaflops, though America, Switzerland, Japan and Saudi Arabia also have (slower) petaflop machines.

Extremely fast computers are used to do aerodynamic computations to reduce air resistance in 18-wheelers; to compute Bitcoins; to forecast the weather a week in advance; to design nuclear weapons; to model the motions of molecules as they engage in chemical reactions (for example, to design more effective drugs or to find genetic components of disease).

And, of course, computer gaming, Figure 61. As the author understands these things, the faster the graphics processor, the better a computer game looks. Major game manufacturers are happy to oblige, and some of the most sophisticated computer design and engineering now goes into game processors. Figure 62 shows Microsoft’s 2017 processor, running an astounding 6 petaflops. Personally, the author wasted an hour trying to get Lara out of the first puzzle, and has permanently sworn off games.

We mentioned floating point operations on a computer. We’ll look at computer numbers in Section 8; most modern computers use a different notation, IEEE 754 normalized floating point notation. A number like 3.145 would be written $3145 \times 10^{-3}$; the number .0023 is $23 \times 10^{-4}$.

Each of scientific, engineering, and IEEE754 put the decimal point in a different position. It might be nice if there were only one correct way to write real numbers, but — in real life, it all depends on the application.