Uncountable, Uncomputable, Unpredictable

Anuj Dawar

22 November, 2016

> The laws of nature are written in the language of mathematics...
>
> Galileo Galilei

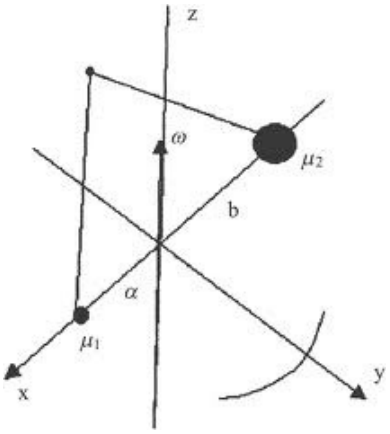
One reason for doing *mathematics* is that this is the language in which theories of *physics* are formulate. Physical theories provide us with a methematical model of the world. One aim of having such theories is that it enables us to make *predictions* about the world. Using a theory for making predictions invariably involves *computation*. But, not everything is predictable or computable.

1 Determinism and Predictability

Newtonian mechanics gives a theory in which the initial position and velocity of all particles in the system completely *determine* the future evolution of the system.

In contrast, in *quantum mechanics* the state of a system only determines the *probability* of measurement outcomes.

Even when the outcomes are determined, it does not mean they are necessarily *predictable*. Prediction requires computation, and not everything is computable. There are fundamental limitations to what can be computed, that have nothing to do with limits of current technology. They lie in the nature of the problems themselves.



The three-body problem consists of determining the evolution in time (according to Newton's laws of motion and gravity) of a system consisting of three point masses, given their mass, initial positions and velocities.
The system is completely described by 9 equations.

It is determined, in the sense that for any given initial configuration and any time t, the position and velocity of each of the particles can be obtained from the equations.

A three-body system (depending upon the initial conditions) can be *chaotic*. That is to say that small perturbations in the initial conditions can lead to arbitrarily large differences in the outcome.

To be precise, for every ϵ, no matter how small, and every Δ, no matter how large, we can find an initial configuration of three bodies so that changing this by no more than ϵ leads to a change in the outcome greater than Δ.

Why is this a problem? While physical quantities like position and velocity vary on a continuum, we can only know them up to some finite accuracy. In short, we don't know a value x, but a range $[x - \epsilon, x + \epsilon]$.

- Say a system is *determined* if, given an initial state s at time 0, the state of the system at time t is fixed to some $F^t(s)$.
- Say a system is *predictable* if, for any ϵ, there is a δ so that, if s is in the range $[x - \epsilon, x + \epsilon]$, then $F^t(s)$ is in the range $[F^t(x) - \delta, F^t(x) + \delta]$.

In this sense, the three-body problem is determined but not predictable.

2 Predictability and Computability

Systems could be predictable but still not computable. What does this mean? Recall that the *natural numbers* are $1, 2, 3, \ldots$, while the *real numbers* are given by (possibly infinite) decimal expansions.

Say a real number x is *computable* if there is a computer program which, when given a natural number i as input, gives as output the ith digit of x.

Not all real numbers are computable. Why?

All rational numbers are computable. Why?

$\sqrt{2}$ and π are computable. Why?

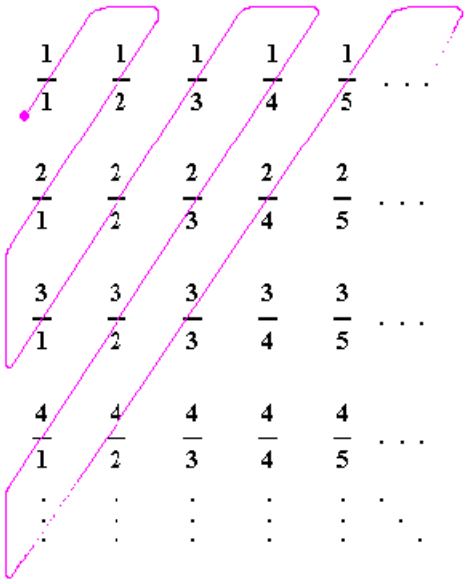
Cantor showed that there are different kinds of infinity. In particular, he showed that there are more real numbers than there are natural numbers. He proved this by a *diagonal argument*.

We say that a set A is *countable* if it can be placed in one-to-one correspondence with the natural numbers.

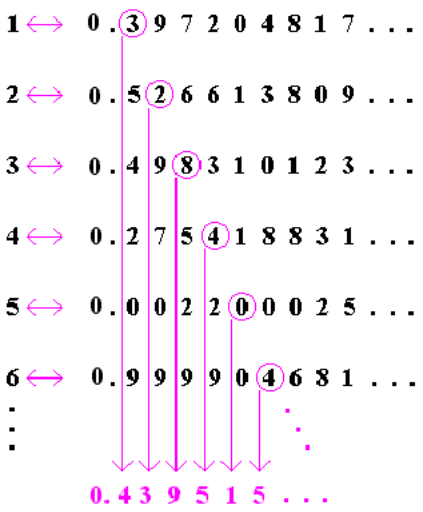
$$\begin{array}{cccc} 1, & 2, & 3, & \ldots \\ a_1, & a_2, & a_3, & \ldots \end{array}$$

The rational numbers are countable. Why?

The real numbers are *not* countable. Why?



The countability of the rationals can be best illustrated by putting all pairs of natural numbers on a quarter plane.
We can then traverse all pairs by taking the finite diagonals in turn.
Note that this repeats each of the rational numbers many times, but this does not pose a problem. The argument actually shows that the collection of all pairs of natural numbers forms a countable set.



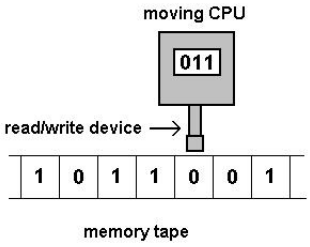
Suppose the real numbers were countable. Imagine then, an infinite list of numbers giving all the real numbers between 0 and 1.
We now define the real number d whose ith digit is obtained by taking the ith digit of the ith number in the list and adding 1. If the digit is 9, we change it to 1.
The number d is a real number that is different from *every* number on our list. This means the original list did not contain all the real numbers.

2.1 Programs

A computer program is given by a finite sequence of symbols. The collection of finite sequences of characters in any fixed finite alphabet forms a countable set. Why?

Thus, there are more real numbers than there are programs and, therefore, there are *uncomputable* numbers. Indeed, most real numbers are uncomputable.

Does this depend on the choice of programming language? Could it be that numbers which are uncomputable in one programming language are computable in another? Not really. Alan Turing showed that there is a *universal* computing machine.



A *Turing Machine* consists of a processor (with a finite memory) and an infinite tape.
At any point, the processor can look at one symbol on the tape.

It contains a program which tells it, given the symbol it is reading, and the contents of its finite memory,

- with what symbol to replace the current tape symbol;
- how to update its finite memory; and

- in which direction to move the tape one step.

Turing also showed specific real numbers that are not computable. Since the set of Turing machine programs is countable, we can fix some enumeration.

$$P_1, P_2, P_3, \ldots$$

of *all* programs.

Now, define the real number $H = 0.h_1h_2h_3\cdots$ by the following rule:

$$h_i = \begin{cases} 1 & \text{if } P_i \text{ halts when given input } i \\ 0 & \text{otherwise} \end{cases}$$

H is *uncomputable*. Suppose it were computable. Then, there is some program Q which, given a number i as input, gives h_i, the ith digit of H.

We define a new program Q' which works as follows:

```
on input i run the program Q;

if the output is 1, then start an infinite loop;

otherwise, stop.
```

Now, Q' is a program, so it appears somewhere on our list. That is, $Q' = P_j$ for some j. What happens when we run Q' on input j?

The proof actually shows that there is no computer program that takes as input programs and tells us whether they contain *infinite loops*. This has been used further to show many other mathematical problems undecidable.

Diophantine Equations:

Given a polynomial equation in several variables:

E.g. $x^5 + 2x^2y + 5xyz = 0$

determine whether or not it has an integer solution.

3 Conclusion

Physical theories provide us with models of the world. One (though not the only) purpose of these models is to enable us to make *predictions* But, the models may not be *deterministic*. Even when they are, they may not be *predictable*. Even when they are, they may not be *computable*, or they may be computable but not *feasible*.

Still, physical theories and computing power are used to make extraordinary predictions: from placing satellites in orbit to manipulating single atoms.