The Long, Long Road to Lattice QCD

Preface: Uselessly Beautiful

The desperate belief motivating the last four centuries of physics research (and, arguably, many other academic pursuits) is that our unfathomably messy reality is generated, at its most basic level, by a relatively simple set of rules.

Incredibly, it seems like this is actually *true*: our little universe, with all of its fish, dirt, mortgages, and supernovae, seems to consistently obey a set of rules far simpler than the emergent structure of its constituents. And through incredible effort, humanity now knows these rules, not perfectly, but well enough to confidently predict isolated physical phenomena with remarkable accuracy.

Yet our stargazing progenitors can't claim a total epistemic victory. Knowing the rules, it turns out, is not nearly enough. The most advanced meteorologist, with all their PRIMITIVE EQUATIONS and rigorously tested theory, cannot tell you what the weather will look like in a month.

Unless, of course, they have a computer.

Accessible Computation

Teleologically, computational physics refers to strategies for spinning up a 'universe in a box'. While this remarkable power is not necessarily helpful for the *development* of physical theories, it is extraordinarily useful when attempting to *test* them. Atmospheric physics, electrodynamics, chemistry, particle physics, solid state — nearly every branch of physics has seen immense gains through the use of **numerical models**: universes-in-boxes designed to explore the emergent properties of physical theories that would be otherwise impossible to study.

To this day, I am fascinated by these universes-in-boxes. There is a perverse joy in the idea of writing a few lines of code, pressing 'run', and watching a complex microcosm emerge from it. The most amazing part is that doing this, achieving that satisfaction — it *isn't even that difficult.* Programs exhibiting interesting emergent phenomena are surprisingly dense in Turing-space, and as I'll show in the first chapter, even the simplest of computer programs can generate strikingly complex behavior worth exploring.

After that introductory sampling, we'll restrict our attention to the subset of microcosm-generating programs that are decent models of reality: physical simulations. As this guide wanders on, we'll focus less on realistically implementing these models, and focus more on their behavior in the infinite-compute limit (though I'll still provide code wherever possible). To guide our study, we'll work towards the titular, most fundamental model of the universe

available to us: **Lattice Quantum Chromodynamics** (**Lattice QCD**). I, admittedly, have never written an implementation of lattice QCD, so I will be learning along with the reader. Consequently, this book will be informal, anti-rigorous, painfully pragmatic, and possibly even incorrect at times (though I will do my best to avoid this).

But I don't really care. I'm not writing this for a journal. I'm writing this for my fifteen-year-old self, who had just written a Barnes-Hut N-body simulator and wanted to keep simulating things. I had no one to turn to back then, no resource to tell me "you should try doing this, next". This guide is intended to fill that gap, and hopefully make it so others can cross it in less than a decade.

1. The Joy of Emergent Systems

I've mentioned that "knowing the rules is not enough", that computers are somehow a necessity for the study of any reasonably complex model. It might not be intuitively clear why this is the case, so I'll take a minute to work through some examples (ordered by ascending complexity) where computers reveal a hidden, complex structure hiding in an innocuous ruleset.

Emergent System 1: L-Systems

Consider the following instructions:

```
1. Replace "R" with "RL"
2. Replace "L" with "R"
Note: both rules are always applied at the same time.
```

For example, if we apply this procedure to the word "PARTICLE", we are left with "PARLTICRE". Similarly, "LIAR" becomes "RIARL", and "JAR" becomes "JARL".

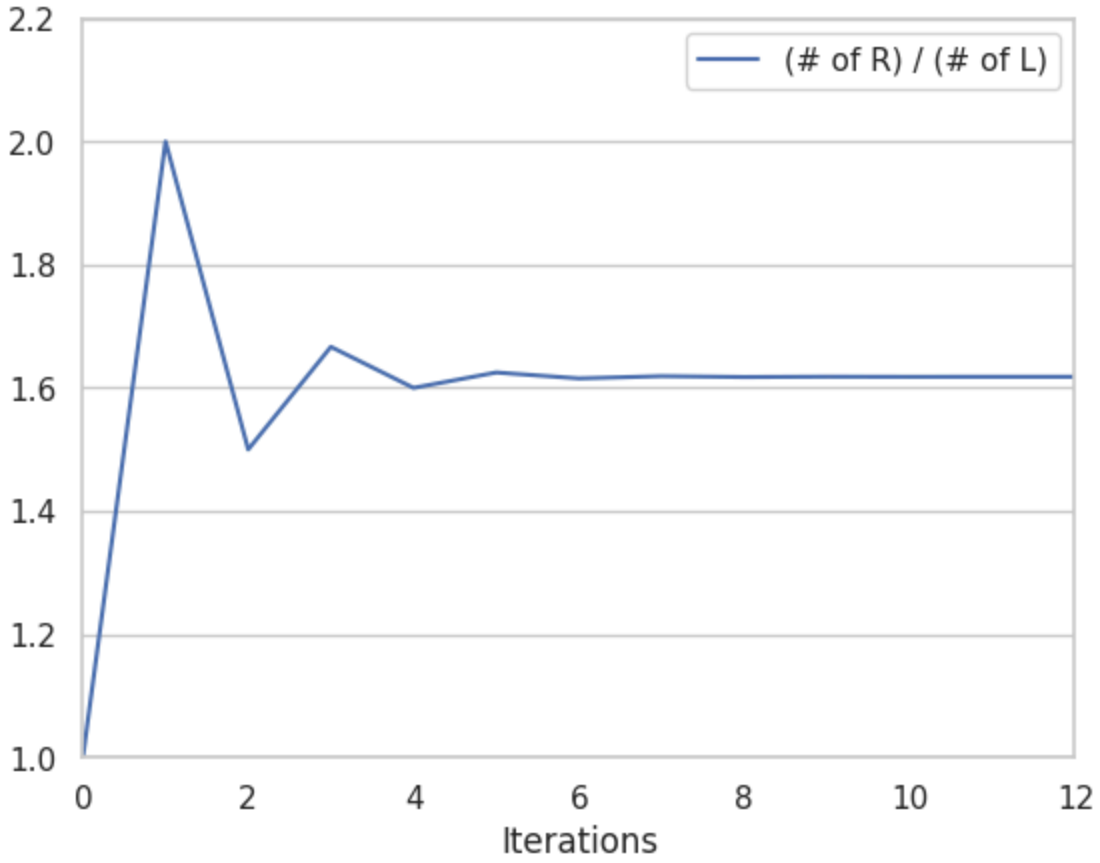
In isolation, not only is this procedure nonsense, it is *boring* nonsense. There is nothing complicated or mysterious about it, and you would be hard-pressed to contrive any interesting intellectual questions from it alone.

But suppose we apply the procedure *repeatedly*, starting with just "R":

```
RL
RLR
RLRRL
RLRRLRLR
RLRRLRLRRLRRL
RLRRLRLRRLRRLRLRRLRLR
RLRRLRLRRLRRLRLRRLRLRRLRLRRLRLRRLRRL
```

```
RLRRLRLRRLRRLRLRRLRLRRLRRLRLRRLRRLRLRRLRLRRLRRLRLRRLRLR
...
```

Okay, this is *especially* nonsensical now, but it's a little less boring. The latter strings in the sequence have some sort of pattern, but it's not as straightforward as you might initially assume. Furthermore, look at what happens to the ratio of `R` to `L` as this process continues:



Slowly, the R:L ratio approaches 1.61803... — ϕ, the "golden ratio". Somehow, this incredibly simple action had an irrational number hidden inside!

Perhaps this isn't especially surprising or noteworthy; irrational numbers aren't especially difficult to construct, and the logic of this rewriting system is tightly bound to the Fibonacci sequence (specifically, our "RL" strings are Fibonacci words).

Nonetheless, somehow, a remarkably messy, complicated structure blossomed from the repeated application of a very simple rule. Hold on to that sensation.

Emergent System 2: The Logistic Map

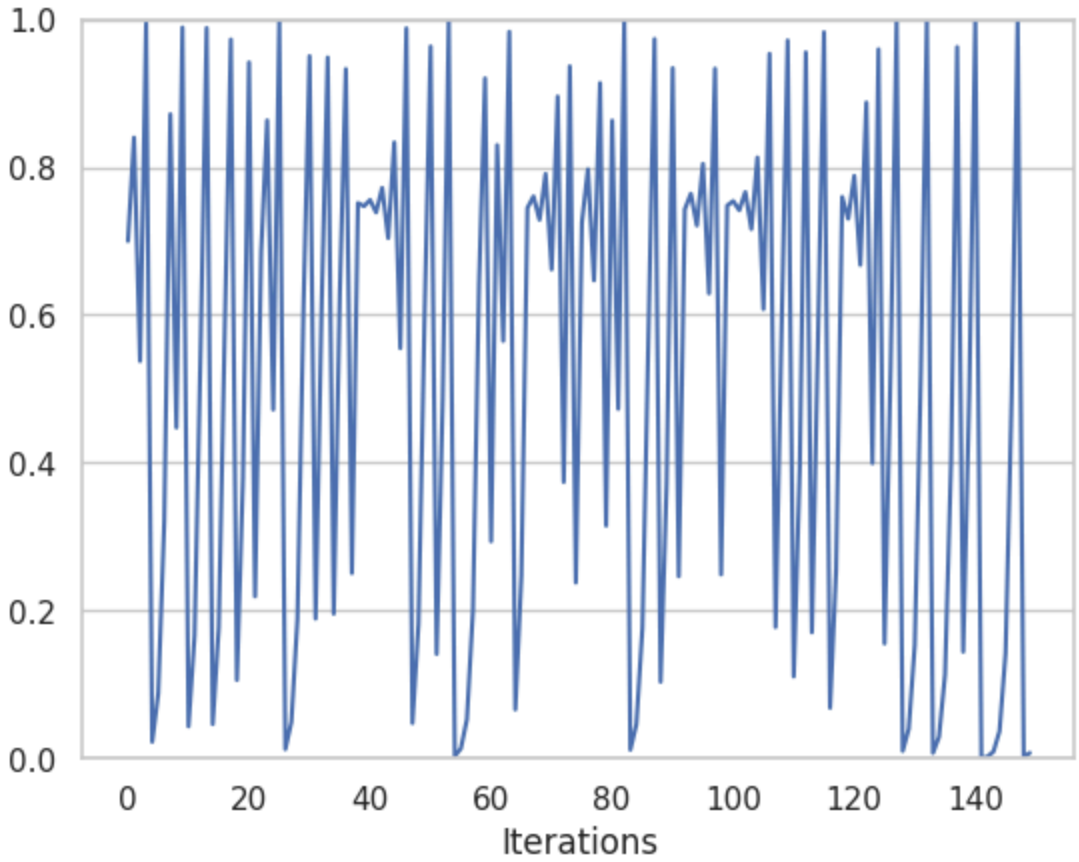
In line with the previous recursive example, please pick a number between (but not including) 0.5 and 1. Now apply the following rule:

```
Your new number = 3.9 * your number * (1 - your number)
```

I'll pick 0.7 for you (sorry), and apply this rule repeatedly, generating the following sequence:

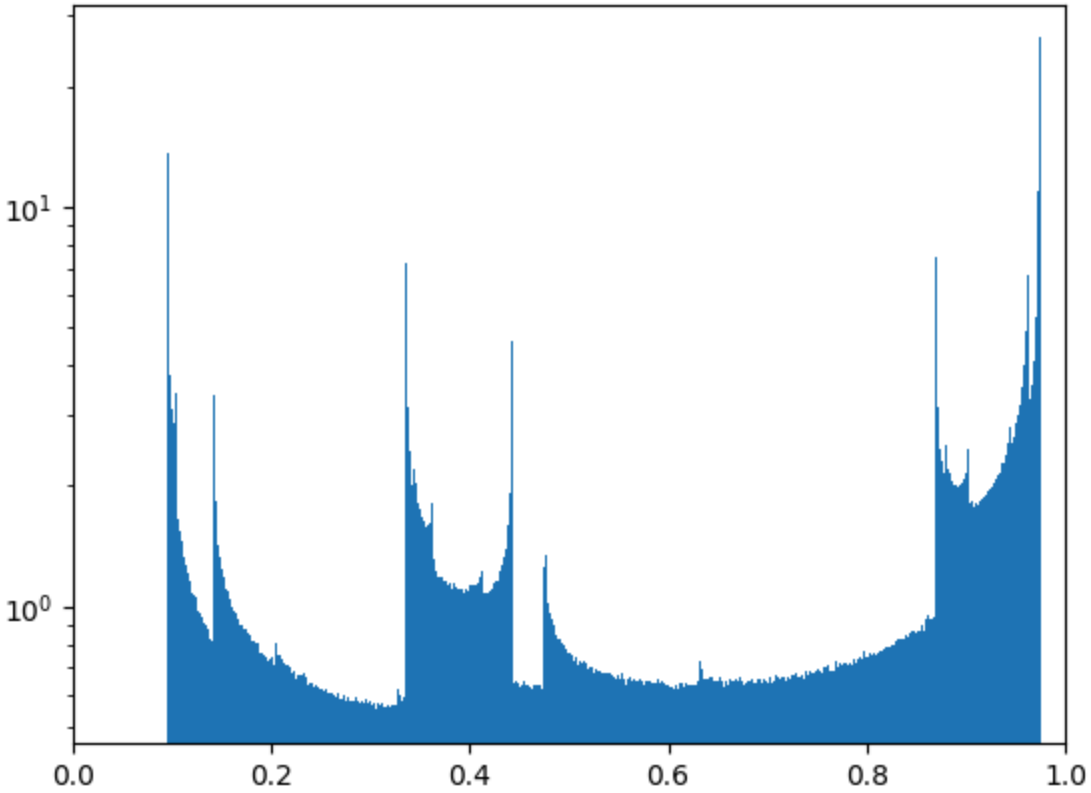
```
0.819
0.5781321
0.9511920
0.1810607
0.5782830
0.9510999
0.1813847
0.5790887
0.9506054
0.1831236
...
```

There is no pattern here. There are recurring *features*, as you can see in the plot below:

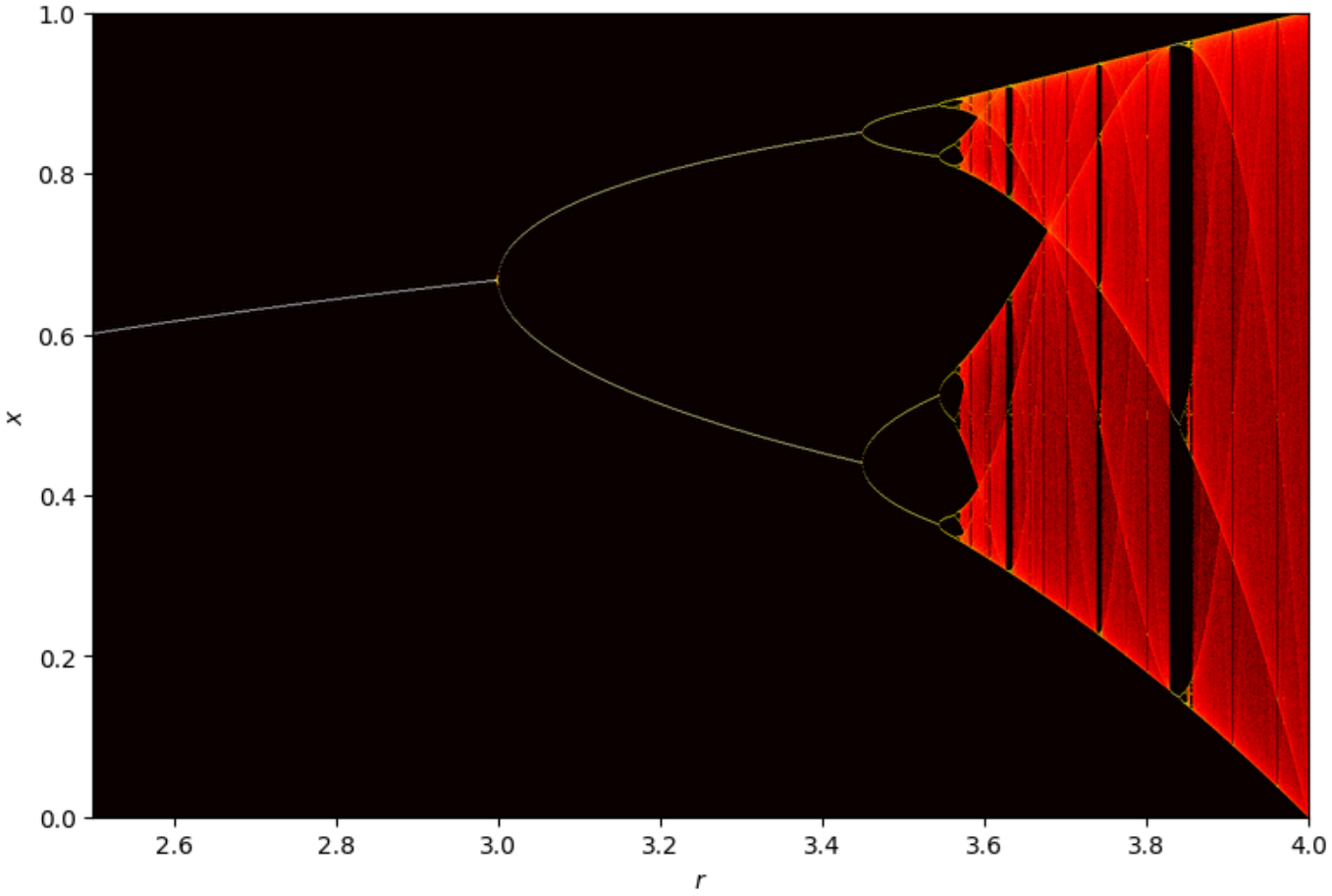


But there is no real *timing* to these features. The system's behavior is erratic, spurious, and unpredictable except by explicitly calling the function itself. We call systems like this **chaotic**. This word means a few different things, but it mostly "hypersensitive to initial conditions"[1]. In this example, if we had started with 0.701 instead of 0.7, the last number in the list above would've been 0.55 instead of 0.18.

Furthermore, if we squish the plot above into a histogram[2] by binning it along the vertical axis (and adding a few million more points), we get this distribution:



We can reveal a more beautiful structure by letting our original value, 3.9, vary between 2.5 and 4.0. This results in a spectrum of histograms like the one above, which, when plotted together, [3] look like this (bin count interpreted as color):



This figure is called a **bifurcation diagram**, for the obvious reason.

Remember, the equation governing this plot is simply $x \mapsto rx(1-x)$, and yet this deceptively simple deterministic process was sufficient to produce this strikingly complex and erratic behavior.[4]

Emergent System 3: Rule 110

Both examples above study *iterated self-maps*: the first map transformed a string into a string, and the second took a number between 0 and 1 and turned it into a different number between 0 and 1.

Rule 110 is another self-map, operating on the space of infinite binary strings (e.g. ...100101101...). We'll view this space as a sequence of blocks, extending infinitely in both directions, filled in for 1 and empty for 0:

---

1. Hypersensitivity alone is not sufficient for a chaotic system: $x_{n+1} = 2x_n$ is sensitive to initial conditions, but clearly not chaotic. Other properties, e.g. **toplogical transivity** of the update function, help refine the definition. ↩
2. The more common choice is to plot a bifurcation diagram; a series of histograms for different values of r ($r = 3.9$ here) plotted as flush vertical lines with counts interpreted as colors. This is certainly a pretty

picture, but perhaps somewhat less interpretable and somewhat more dismissive of the strange jumps and gaps present in the map's histogram. ↩

3. If you're having trouble interpreting this figure, the previous histogram (where $r = 3.9$) looks like a vertical line on this plot, where the value along the x axis is 3.9. ↩
4. A complex extension of the logistic map yields the Mandelbrot set, an iconic figure in the study of fractals and chaos. ↩