Randomized Periodic Walks

Harshit Chhabra

Introduction

There exists a myriad of phenomena in nature that beget a comprehensive study of complex systems containing a large number of particles. While I had some previous knowledge, I recently came across this discipline through an insight while studying a phenomenon called Group Polarization[1] in Social Psychology. Speaking in very broad terms, I realized that in physics, while working with a large number of particles, you can develop accurate models without having precise knowledge of motion of each individual particle in the system, the so-called Statistical Mechanics[*] While I'm aware that my knowledge in both of these domains, Social Psychology and Statistical Physics is very limited, and the Analogy itself might not make much sense, it was enough for me get motivated about the study of complex physical systems. Furthermore, I realized that systems like an orchestra consist of numerous instruments playing together to form complex notes and tunes, however if we look at the individual notes being played by a single instrument, say a violin, it usually consists only of simple (or a simple combination) repeating tones and notes. I realized that individual particles can evolve in their own ways, for example, simple and periodic motions. I wanted to see what would happen to the overall orchestra if we just varied a singular or a couple instrumentals by a random factor.

Setting up the Simulation

To simulate a complex system[†], I took the help of computational tools, mainly Python 3, and later on the Quantum Computing Module for Python 3 by IBM, Qiskit[2] . I started by initializing a simple float type list, containing *n* elements (n = 1, 2, 3…and so on, determined by the user as input). The user would also input a starting value $x \in \mathbb{R}$ and the number of iterations, *k* (integer ≥ 1). Then, the program[3] would initialize a list containing n elements with each element a_j of the form,

$$a_j = \sin(x + j + l),$$

Where, $j = 0, 1, 2\ldots, n$

l is a variable which will increase every iteration,

like, $l = 0, 1, 2\ldots, k$

This ensures a periodic property not only to subsequent elements in a list but also to elements at same index (*0 to n-1)* in subsequent iterations. This way, as the system runs for the given number of iterations, it evolves in an independent periodic fashion. We can now introduce a randomized variable, *ifactor* (initialized to 0), and rules for how it will affect our system.

We will use the Arithmetic Mean *M where,*

---

* I would like to point out that the original insight was that, like in Group Polarization, the individual properties sometimes can be left out while studying complex systems. We can use statistical methods to model the system without considering individual properties of each particle. This property, seemed very similar (to me at that time) to the apparently 'simpler process of extreme decision making' in Group Polarization. Given a large enough group, the individual values and opinions can be left without much consideration, at least that was my understanding at the time when I had the insight, there could definitely be other nuances and subtleties.

† I should point out here that I used the tools and knowledge I had at my disposal. This paper is not claiming to set up a good representation of a complex system. I wanted to simulate a periodically evolving system with a random factor and study the effects. And that is what I have attempted to do. I am still learning and with more knowledge this system can improve a lot.

$$M = \frac{\sum_{j=0}^{n} sin(x+j+I)}{n}$$

of all elements in the list in any given iteration as the Central Tendency to be studied. *M* for a particular iteration will be calculated after accounting for the ifactor and precision.

Randomized Variable, *ifactor* and Qiskit

Every iteration, there is a certain probability, *P* (it was set to *0.034%* but can be easily changed) that the *ifactor* will change by a number in the Integer Range [-50, 50]. Then all the elements in the list for that iteration, would change by *i = ifactor/n.* M_j (*j* being the iteration) is then Calculated for this Iteration and added to a list containing *M* for every iteration, which is then plotted in a graph, *M vs k.*

The *ifactor* is analogous to the random variation in a single instrument in the orchestra. It is supposed to simulate Random Excitations and De-excitations gas particles in a box may undergo.

In earlier versions of the program, the variable *ifactor* was randomized using Python's Random Module, a Pseudo-Random Number Generator[4] (PRNG). However, later I utilized the Quantum Computing Techniques of Qiskit to decide on a number for the *ifactor*. This Change does not serve any beneficial purpose as far as I know. However, another study could naturally come out from this experiment: The Difference between the inherently random Quantum Computing and PRNGs, (we may have to assume arbitrary precision, which implies that for all our purposes, both are equally useful).

I utilized the Hadamard Gate Operation[5] to prepare states with equal probability and run them through a code which will measure and output the state with the highest occurring count.

Figure 1: Sample Data

```
iteration:  11
central tendency:  1.5137
1.1476 0.7001 1.1719 2.1292 2.6919 2.3427 1.4024999999999999 0.7359 0.9557 1.8598

iteration:  12
central tendency:  1.6607
0.7001 1.1719 2.1292 2.6919 2.3427 1.4024999999999999 0.7359 0.9557 1.8598 2.617

iteration:  13
central tendency:  1.8438
1.1719 2.1292 2.6919 2.3427 1.4024999999999999 0.7359 0.9557 1.8598 2.617 2.5311

ifactor:  44
iteration:  14
central tendency:  4.5947
4.8292 5.391900000000001 5.0427 4.1025 3.4359 3.6557000000000004 4.5598 5.317 5.2311000000000005 4.3811

iteration:  15
central tendency:  4.4666
5.391900000000001 5.0427 4.1025 3.4359 3.6557000000000004 4.5598 5.317 5.2311000000000005 4.3811 3.5485

iteration:  16
central tendency:  4.2773
5.0427 4.1025 3.4359 3.6557000000000004 4.5598 5.317 5.2311000000000005 4.3811 3.5485 3.4987000000000004

iteration:  17
central tendency:  4.2008
4.1025 3.4359 3.6557000000000004 4.5598 5.317 5.2311000000000005 4.3811 3.5485 3.4987000000000004 4.2776000000000005
```

Notice the discrepancy in the significant digits despite setting the precision to 4 digits after the decimal point. I will elaborate on this particular point later in this paper

Figure 2.a

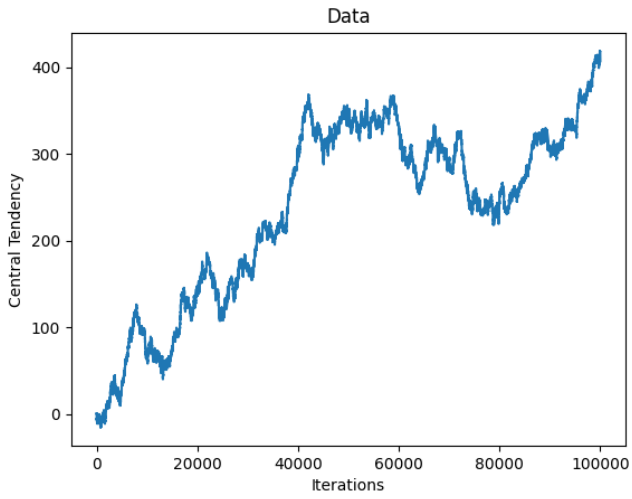

Figure 2.b

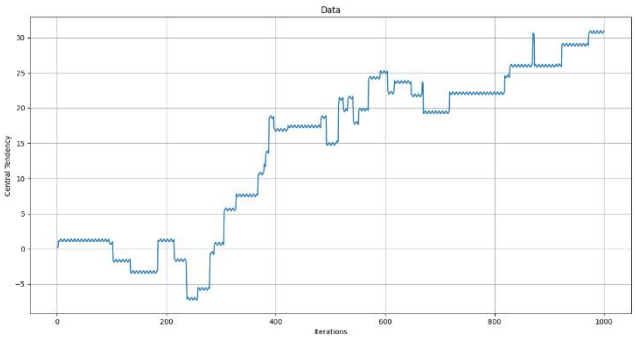

Figure 2.a : An M vs k graph plotted for 100,000 iterations with x = 0 and n=10 ; Figure 2.b : A graph plotted for fewer iterations (1,000) to reveal the periodic nature of the graph. I highly recommend checking the GitHub repository (linked below) as more details about the output can be found there

Random Walks

As I was writing the code for the simulation, I realized that this particular system, given the way it's being plotted and studied, is similar to a very powerful concept, Random Walks[6].

As the simplest example, imagine a flee on the Integer number line. Every second, the flee can either hop 1 unit distance to the right (+1) or 1 unit Distance to the Left (-1) with equal probability for both (50%). The description of the path taken by the flee is the simplest set up of a Random Walk.

In other words, a particle has an equal probability of moving by a fixed unit distance per second (or each iteration) in a One-Dimensional Mathematical space.

I will not go into the details[‡] and the math, but we can generalize this system to a 'hop' of N-units distance on either side (or even in multiple dimensions) with varying probabilities (the sum of all probabilities, however, should be 1). Coming back to example of the flee, now it may move by N-units distance in 3 dimensions with a certain probability for each direction (up, down, left, right, front, back).

(Note that the flee can start from anywhere, not just the origin)

This simple notion of a Random Walk is used everywhere, from casinos to finance and is even very helpful in studying certain particular complex systems in Statistical Mechanics (Ideal Gas, Kinetic Gas Theory[7])

[‡] However, I highly encourage the reader to use some of the links I will provide (and some you may find on your own) and some other illustrations available online. It is a truly fascinating topic and if I pursue this project further, it will surely be used a lot. But as of right now, we only need some simple familiarity of the concept.

Similarity Between the Experiment and Random Walks[§]

In the simulation we've been studying, we plot the central tendency of each iteration, after accounting for *ifactor.* This implies, we are, effectively (mostly because of the choice of the Central Tendency and rules on *ifactor*) plotting the randomized change in a singular value with some periodic property (The singular value being the Arithmetic Mean, *M)*

(Using our example of the flee, we can say that every iteration, the flee will move a certain unit distance regardless of the ifactor. When the ifactor ≠ 0, then the distance moved by the flee is changed according to certain rules. Hence, the flee in this case would be the Arithmetic Mean, M)

In other words, we can say that that a singular point of data, *M* has a certain probability that its value changed by a random number (in the range [-50/*n* , 50/*n*]). It's clear now that this particular system and the way it's being plotted is similar to the functioning of a Random Walk.

Moreover, as the number of elements in the list, *n* gets very large, the average of that list would approach 0. This is because the list in each iteration is simulating a sine wave, and the average of all the values of the sine wave is 0. And as *n* tends to infinity, the list will cover more and more of the sine wave, and *M* will tend to 0.

This simple fact implies that if we ignore the *ifactor* for a while and keep increasing *n*, then the graph *M vs k* will become approach the constant value 0. When we re-introduce the randomized variable *ifactor,* the system will imitate a random walk, without the periodic feature, instead it would be a constant line, *M = 0* during iterations where the *ifactor* is 0.

This similarity implies that we can develop a mathematical model for this particular simulated system using the same mathematical tools used to describe Random Walks. It also implies that we can develop this model for Higher Dimensions, with the *ifactor* now being able to 'shift' or 'rotate' the periodic data source by some value into other directions. Although extensive study is required by myself to be able to achieve that both computationally and mathematically.

Possible Errors, Future Opportunities and some Observations and Deliberations

1) The similarities between Random Walks and my models may solely be due to the fact that I used Arithmetic Mean to study the system, effectively reducing the periodic function to a single quantity. That would mean whatever I am doing to each element in the list, I could do the equivalent to the Mean to achieve the same result. This does not necessarily mean that the model is identical to Random Walks. It just implies that my model is simulating systems which have similarities to Random Walks. I am interested in seeing the results when we use different measures of Central Tendency, like deviation from the RMS of the data.
2) If using different plotting techniques and Measures of Central Tendency yields different results, that would imply that we can use this model as a template to study various types of systems by changing the measures of central tendency. That is something that I am interested in as well.
3) Bear in mind, I haven't yet described the model mathematically. While it is a very simple model, and we can describe its evolution completely just by knowing the starting value *x,* I would like to show the resemblance to Random Walks after introducing the *ifactor* using

[§] Out of sheer coincidence, I stumbled upon a mathematical object which seemed very familiar to my simulation. The object is something called the 'Cellular Automation'. It was an interesting find and even though I will not tough upon that topic in detail, it could be of use in the future.

mathematical methods. I'm hoping this would lead to more insights into these types of models and open more future directions.

4) From the start, I was interested in systems where individual 'particles' are inter-related in some way. I tried to simulate this property by a) simulating only one parameter that b) is related to the rest of the particles through the sine function. I am interested in looking for other (potentially more complex and intricate) ways to simulate periodically evolving intra-connected systems and introducing a randomized variable.
5) Another reason for this model's resemblance to Random Walks is due to the rules imposed on the random variable, *ifactor.* The effect of this variable is homogenous throughout the list of data in the given iteration. This means that the Mean is changed by a constant, which adds to the similarities with Random Walks. I am very interested in changing how the *ifactor* affects the system and observing the resulting changes. A simple example would be that in a system of densely packed particles, any excitation in any particles would affect the direct neighbours more and the far neighbours less, instead of a homogenous change. Hence, we can build a simulation with the *ifactor* affecting particles as a function of their position w.r.t randomly chosen particle for the excitation.
6) In the sample data, we can notice the extra number digits appearing after the decimal point, despite setting precision to *4 digits after the decimal*. In some simulations I ran, it was noticed that sometimes these extra digits would 'accumulate' over iterations and sometime affect the accuracy of the data in subsequent iterations when they were finally rounded off to the set precision. This was seen in the graph as 'blips' in the iterations where the *ifactor* was 0 but the point on the graph was off by some value. I would also like to study potential chaos in such systems, stemming from the approximation techniques.

   (There is a possibility however, that this could be nothing as this seeming 'chaos' generally resolves itself over the next iterations after the supposed 'blip'. However, I suspect this is because I create a new list every iteration, so everything is new except the ifactor. We need to create a specific model to 'catch' this chaos or at least study the graphs more carefully)

7) While studying some topics for another project (in theoretical physics), I cam across the mathematical concept of a Cellular Automaton[8]. I would certainly like to study this concept for any similarities to what I'm attempting to do and maybe even apply some methods of my simulation to this concept.

References

1) Group Polarization - Cass R. Sunstein, The Law of Group Polarization (John M. Olin Program in L. & Econ. Working Paper No. 91, 1999).
2) PRNGs - Koeune F. (2005) Pseudo-random number generator. In: van Tilborg H.C.A. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA . https://doi.org/10.1007/0-387-23483-7_330
3) Random Walks - **Stephanie Glen**. "Random Walk" From **StatisticsHowTo.com**: Elementary Statistics for the rest of us! https://www.statisticshowto.com/random-walk/
4) Kinetic Theory of Gases - Maxwell James Clerk 1867IV. On the dynamical theory of gases *Phil. Trans. R. Soc.***157**49–88. http://doi.org/10.1098/rstl.1867.0004

Links and Sources

[1] http://nrs.harvard.edu/urn-3:HUL.InstRepos:13030952

[2] https://qiskit.org/ ; https://github.com/Qiskit ; https://developer.ibm.com/depmodels/quantum-computing/projects/qiskit/

[3] *My GitHub repository, containing the codes and some more updates and data*: https://github.com/harshit51-blip/Randomized-Experiment

[4] https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=Pseudo%20Random%20Number%20Generator(PRNG)%20refers%20to%20an%20algorithm%20that,state%20using%20a%20seed%20state ; https://doi.org/10.1007/0-387-23483-7_330

[5] https://www.quantum-inspire.com/kbase/hadamard/

[6] https://www.statisticshowto.com/random-walk/ ; https://www.math.ucla.edu/~biskup/PDFs/PCMI/PCMI-notes-1 ; https://towardsdatascience.com/random-walks-with-python-8420981bc4bc

[7] http://doi.org/10.1098/rstl.1867.0004 ; https://en.wikipedia.org/wiki/Kinetic_theory_of_gases

[8] Cellular_Automaton_Wolfram