THE UNIVERSITY *of* EDINBURGH
School of Physics
and Astronomy

Senior Honours Project

Ergodicity of a System of Three Hard Spheres on a Ring

Matej Vedak
April 2022

Abstract

We investigate ergodicity of three pointlike particles on a ring, conducting elastic collisions. A novel method of exploring ergodicity is used, which consists of transforming the particles' collision history into a sequence of binary digits. Entropy, compression percentage, and randomness of the produced strings are tested and compared against random number generator created strings.

Declaration

I declare that this project and report is my own work.

Supervisor: Professor G. J. Ackland 10 Weeks

Contents

1 Introduction

Statistical physics is arguably one of the most important modern day areas of physics. It allows physicists to deduce properties of very complex systems, systems whose equations of motion are not feasible to solve. Nevertheless, its usefulness is determined by the validity of its central assumption: a physical system must sample a representative area of its theoretical phase space. This leads to the question of the ergodicity of a system. Points in a moving ergodic system will eventually visit all parts of the phase space that the system moves in. Ergodicity captures every-day notions about randomness, such as that released smoke in a room will come to fill out the entire room, given enough time. These systems can be tackled using methods of statistical physics. Ergodicity implies that the average behaviour of the system can be deduced from the trajectory of a 'typical' point. Equivalently, it means that taking a sufficient amount of random samples of the process can represent the average statistical properties of the whole process. However, it is not clear that a deterministic system will visit all available microstates [1] [2].

This is why explorations of ergodicity in relatively simple systems became an important area of research. It is of interest to find systems that are complex enough to exhibit non-trivial ergodicity but simple enough so we can actively examine the phase space in which they move. Many such systems have been explored, and of particular importance are hard-ball systems because of their simplicity (for example, [3]).

The system we are studying consists of three pointlike particles that move on a periodic ring. The phase space of this system is six dimensional, with axes consisting of three positions and three momenta of the particles. Conservation of energy and momentum reduce this phase space to a four-dimensional hypersurface within this six-dimensional phase space. Particles continuously explore position space, while coordinates in velocity

space (and by extension momentum space) remain constant until a collision occurs, after which they take on new constant values. Labelling the initial momenta with p_1 and p_2 and the subsequent post collision momenta with q_1 and q_2 of particles with masses m_1 and m_2 the update formulae are:

$$q_1 = \frac{m_1 - m_2}{m_1 + m_2} p_1 + \frac{2m_1}{m_1 + m_2} p_2, \tag{1}$$

$$q_2 = \frac{2m_2}{m_1 + m_2} p_1 + \frac{m_2 - m_1}{m_1 + m_2} p_2. \tag{2}$$

The outcome of every collision can be represented by the operation of a matrix on a vector of momenta (p_1, p_2, p_3), a state vector in the phase space. Any sequence of collisions can be mapped onto a product of such matrices [4].

The spatial probability distribution was found to be the triangular probability distribution for each of the three particles, going to zero where a triple collision occurs. The analytical derivation of this result can be found here [5]. Momentum space distributions assume a vastly different shape. As outlined in [4], the shape of three particle momentum distributions is similar to a Lorentz factor function (formally called an arcsine distribution):

$$P(p_i) \propto (p_{i,max}^2 - p_i^2)^{-1/2}.$$

This distribution comes around because it is the projection of an ellipse onto one axis in the 3D momentum phase space (the ellipse being the region that satisfies the conservation of energy and momentum). The mean energy of each particle can be calculated [6]

$$E_i = \int \frac{p_i^2}{2m_i} P(p_i) dp_i = \frac{M - m_i}{2M} E_{tot}.$$

It is proportional to the mass of the other two particles. Departure from the usual equipartition result is due to the fact that the system is in the 'molecular-dynamics' ensemble of statistical mechanics where energy, momentum, volume, and the number of particles are conserved [7] and a generalized version for any number N of particles can be found [4]:

$$\langle E_i \rangle = \frac{\sum_j m_j - m_i}{\sum_j m_j} \frac{E}{N-1}.$$

This system has been shown to be equivalent to a billiard problem in a triangular stadium [8]. With the triangular system in mind, it is easy to see that there are periodic orbits (example of a right triangle where the billiard bounces off the two non-hypotenuse sides at a right angle). Since one cannot escape the periodic orbit, and the dynamics are time-reversible, one cannot enter it either. In this way, the microstates of a periodic orbit remain inaccessible to the general case, which violates ergodicity.

It is essential to study these simple systems to evaluate how closely deterministic chaotic systems approach ergodicity. Explorations of the standard type have already been made [6], where it was investigated how efficiently this system samples the phase space. This paper takes on a different approach. We generate binary strings based on particle collisions and compare those strings to completely randomly generated ones. The more random the collision generated strings end up being, the stronger belief in the ergodicity hypothesis can be held and the system is better at exploring its phase space. To the best of our knowledge, this is a novel approach that has not been explored yet.

2 Methods

All code was written from scratch in python. Usual numerical simulations advance simulations in terms of 'timesteps'. Differential equations are discretized, and the timestep is then the smallest discrete time interval of the resulting numerical solution. The system of 3 particles on a ring is different. There are no external forces, and the system is simple enough that it can be solved 'collision by collision' analytically. That is why, instead in terms of timesteps, simulation was advanced in terms of events, which is called an event driven simulation. Each event corresponded to a collision between two particles. After each collision, time to the subsequent collision and two particles that collide were found.

The simulation is relatively straightforward, apart from three significant problems that had to be resolved:

1. Floating point errors. For the simulation to be precise enough, floating point errors had to be taken into account. They are a feature of every programming language and are the feature of the way computers themselves store fractional numbers. To this end, we employed python's 'Decimal' module. It provides exact arithmetic, free of floating point errors up to a specified precision, at the expense of computation speed. We used the default precision, which is 28 significant digits. More importantly, it allows for exact comparison of floats, which was very important in finding the subsequent collision (for example, had the module not been utilized, the following comparison would have returned False: $0.1 + 0.1 + 0.1 == 0.3$). Computation reduction seems to be around a factor of 6, small enough not to have any significant impact on this project.

2. Even with floating point errors resolved, because of the way the simulation was coded in, events of particles 'phasing' through one another happened. This issue was resolved by noting that particles on the ring always have to have a particular ordering. Ring's coordinates can be encoded into the interval $[0, 1\rangle$, which is periodic. If we initially label the three particles with increasing indices, going left to right, after any number of collisions the value of the Levi-Civita tensor on the resulting indices must remain 1. For example, looking at indices from 'left' to 'right', the following combination can happen "201" while the following means particles phased through one another "210". This allows for the prevention of phasing as the ordering is checked after every collision; if there are discrepancies in the particles that have just collided they are manually switched, so the simulation proceeds correctly.

3. In the unlikely (or preset) event of a triple collision, it is vital that all collisions are detected and respectively collided. Choosing which two particles will collide first is somewhat arbitrary and introduces some chaotic behaviour to the system. In this simulation, particles were always collided such that the pair more to the 'left' collided before the pair more to the 'right'.

After convincing ourselves that the simulation worked as indented, we started generating strings from the collision history of the particles.

Start with three particles. Sorting by their starting position, we index them: $0, 1, 2$. Meaning at the onset of the simulation $x_0 < x_1 < x_2$ ($x_i \in [0, 1\rangle$). Collisions are simulated and say that the following collisions happen: particles 1 and 2 collide, particles 0 and 1 collide, particles 0 and 2 collide, particles 0 and 1 collide. In the datafile, the collisions would be written down as follows:
$(1, 2)$
$(0, 1)$
$(2, 0)$
$(0, 1)$.
Strings were generated in the following way: after two particles collide there are only two options, either the 'left' particle collides with the third one, or the 'right' particle does. A 'left' particle collision is encoded by a 0, collision of a 'right' particle is encoded as 1. Using the above collisions as an example, we will explicitly write out the generated string. The first collision is between particles 1 and 2, and subsequently, particle 1 (on the 'left' of the collision) collides with the third particle, particle 0. At that point, the string we are generating will look like:

$$s = 0.$$

The subsequent collision is between particles 2 and 0; again, 0 was the particle on the 'left' of the previous collision so the string is updated accordingly:

$$s = 00.$$

The final collision in the example is $(0, 1)$, where 0 was the particle on the 'right' in the previous collision, so a 1 is appended to the string:

$$s = 001.$$

These strings are referred to as collision generated strings, as opposed to random strings, which were created using random number generators (RNG, created using python's numpy.random module, they are technically pseudo-random numbers).

Perhaps the most obvious first test to conduct is the inspection of entropy of the strings at hand. Entropy will tell us what the frequency of the occurring characters is. The information theory approach was used, and Shannon entropy was utilized:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2(P(x_i)).$$

In the binary string case, x_0 corresponds to the total proportion of 0s in the string, while x_1 corresponds to the total proportion of 1s in the string.

Furthermore, there are many other ways of randomness testing. In this context, we are interested in statistical randomness. A sequence of characters is statistically random if it does not contain any recognizable patterns or regularities. It is important to note that statistical randomness does not necessarily imply true randomness or objective unpredictability. This is due to the fact that any inspected and tested sequence must be finite. Any tests conducted on such a finite sequence can only prove 'local' randomness, even though we are interested in the true, 'global'. Furthermore, according to Ramsey theory,

sufficiently large objects must contain some sort of substructure [9] (good example can be found here [10]), what in the case of binary strings means that there will be significant compression even for strings generated by a random number generator.

First randomness tests were published by M. G. Kendal and Bernard Babington Smith [11] and originally consisted of four tests:

- The frequency test checked whether the number of 0s, 1s, 2s, and other digits was roughly the same.
- The serial test did the same thing as the frequency test, but for sequences of two digits at a time, comparing the observed frequencies with the hypothetical predictions.
- The poker test tested for specific sequences of five numbers at a time based on hands in the game of poker.
- The gap test looked at the distance between zeroes (00 is a distance of 0, for example).

Since then, randomness tests have increased in complexity and have improved. Perhaps the most famous and popular are the so called diehard tests [12], a battery of statistical tests that measure the quality of a random number generator [13].

Another way of testing how random a sequence of characters is is by using a compression algorithms and seeing how compressed the data turns out to be. Furthermore, comparing the compression percentage of collision generated strings with actual RNG strings provides with a meaningful comparison of randomness. This works because the compression algorithms themselves try to exploit repetitiveness and correlations of the given data, and therefore they will create better compression on non-random data sequences.

In this project, we used one diehard test - the Wald-Wolfowitz runs test [14]. It can be used to test the hypothesis that the elements of the sequence are mutually independent. It uses runs - a run is defined as a series of increasing values or a series of decreasing values. The number of increasing or decreasing values is the length of the run. In a random data set, the probability that the next value is smaller or larger follows a binomial distribution, a fact that forms the basis of the runs test. The test's null hypothesis is that the sequence was produced in a random manner. Define the following values: n_1 is the number of positive (increasing) runs, n_2 is the number of negative (decreasing) runs. The expected number of runs is then given by

$$\overline{R} = \frac{2n_1n_2}{n_1 + n_2} + 1,$$

the standard deviation by

$$s_R^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}.$$

The test statistic is

$$Z = \frac{R - \overline{R}}{s_R}$$

where R is the observed number of runs. The runs test rejects the null hypothesis if $|Z| > Z_{1-\alpha/2}$ where α is the desired significance level. For example, if we desire a 5% significance level, the test statistic with $|Z|$ greater than 1.96 indicates non-randomness.

Compression algorithms that were used are zlib [15], bzip2 [16] and a custom run-length encoding algorithm. Zlib and bzip2 are very complex algorithms and we will not go into detail about their inner workings. The custom algorithm works by collecting subsequent occurrences of same characters into a number of occurrences and the character itself. It simply replaces same characters that occur in a series with the number of occurrences and the character itself. For example, the string *aabaaabbb* is compressed to 2*ab*3*a*3*b*.

3 Results & Discussion

We start with the verification of the previously mentioned results for a system of three particles on a ring. Example position and velocity distributions can be seen in figure 1. The triangular distribution in position space and the arcsine distribution in velocity space are nicely depicted.

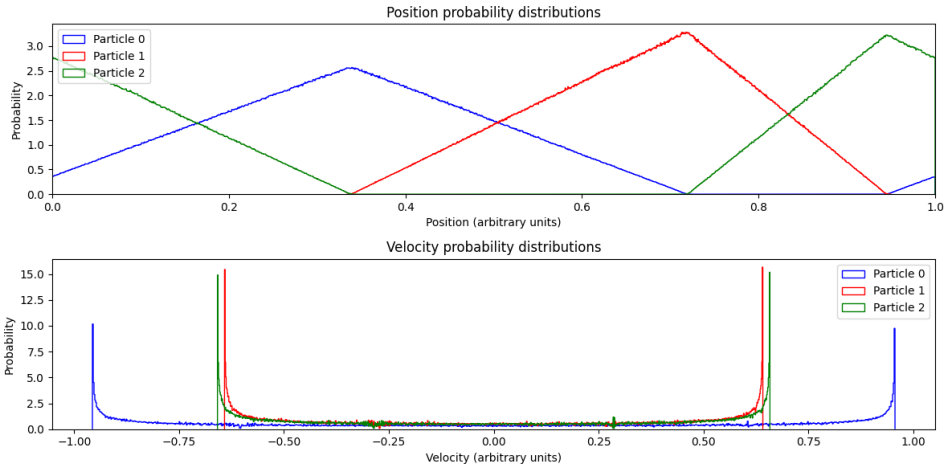

Figure 1. Position and velocity probability distributions for three particles on a ring. Random initial conditions. Position probability distributions follow the triangular distribution and each particle is confined to its region, restricted by points where a triple collision occurs. Velocity probability distributions are of the form $P(v_i) \propto \frac{1}{\sqrt{v_{i,max}^2 - v_i^2}}$ where $v_{i,max}$ is determined by the masses of the particles and the total energy of the system.

Interesting to note is that the smaller the mass of the particle, the more spread out its velocity distribution will be, but its momentum distribution will be less spread out. In Figure 2, we can see that the particle with the smallest mass has the most narrow momentum distribution but the widest velocity distribution. This result comes around because of the conservation of momentum, lighter particles have to acquire higher velocities so that their total momentum can offset the momentum of the more massive particles.

Moving on to string generation, we investigated the entropy and compression percentages of strings with differing lengths. Figure 3 depicts these results. Strings that were generated by particle collisions are depicted in blue, strings that were generated by random numbers are depicted in orange. Results are averaged over 40 runs of randomized initial

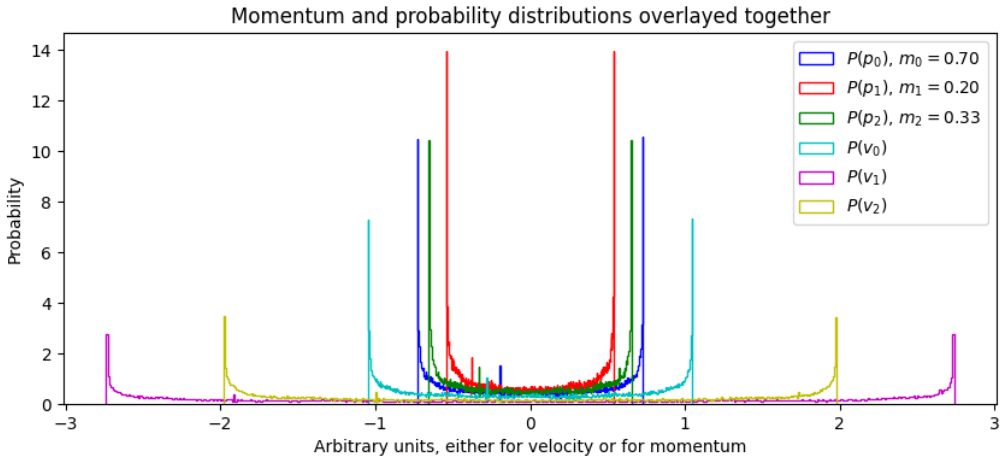

Figure 2. Sample momentum and velocity distribution shown on the same plot. Typical distribution results can be observed - small mass particles have narrow momentum distributions and broad velocity distributions, high mass particles have wide momentum distributions and narrow velocity distributions.

starting conditions (for each string size, 40 random initial condition experiments were conducted, and the results were averaged). Immediately, differences can be observed. The entropy of collision generated strings only stabilises at longer lengths, while random strings take on the value of 1 bit of entropy early on. This is early indication of long streaks of a repeating single digit in the collision generated strings that only tends to even out at longer string lengths. Similar results are observed in the zlib and bz2 algorithms, which exploit these long runs of identical digits and other complex methods to compress collision generated strings substantially better than random strings. A similar pattern is seen in the custom run-length encoding algorithm, where collision strings are compressed by $\sim 70\%$, while random strings only see improvement in size for small strings, and no improvement is found for longer string lengths. Besides allowing zlib and bz2 compression algorithms to utilise their full power, string length has no substantial effect on the compression percentages. Generally, collision generated strings are not as 'random' as RNG strings. Furthermore, RNG strings do not get compressed to less than 16%, both in zlib and bz2 compression algorithms.

These results are not surprising, as it is evident that for some mass configurations of the system (example: all masses the same), the collision generated strings will consist of only one digit for any number of collisions. What is interesting is the fact that a 'general' system is not as 'random' as actual RNG strings. Perhaps certain combinations of masses produce strings that appear more random. To that end, we utilize the Wald-Wolfowitz runs test on strings generated by a variety of mass ratios. Depicted in Figure 4 we can see that the Runs test Z score is quite high and most strings would not be accepted as random binary strings. However, a few regions of low Z scores would be accepted as coming from a random distribution (at least concerning this one test, further testing would be optimal because, as stated above, 'global' string randomness can never be proved).

This region of low Z scores was further investigated (Figure 5). Area of low Z scores was enlarged, and mass ratios that produce a 95% confidence of randomness were found. Additionally, a cut was made along the ratio $m_2/m_3 = 0.87$. The minimum average Z score was found to be 2.08, which is slightly above the 5% confidence value of 1.96. There are slight variations in the final results due to the inherent chaotic properties of the system. Different initial starting conditions (positions, velocities) will give out similar,

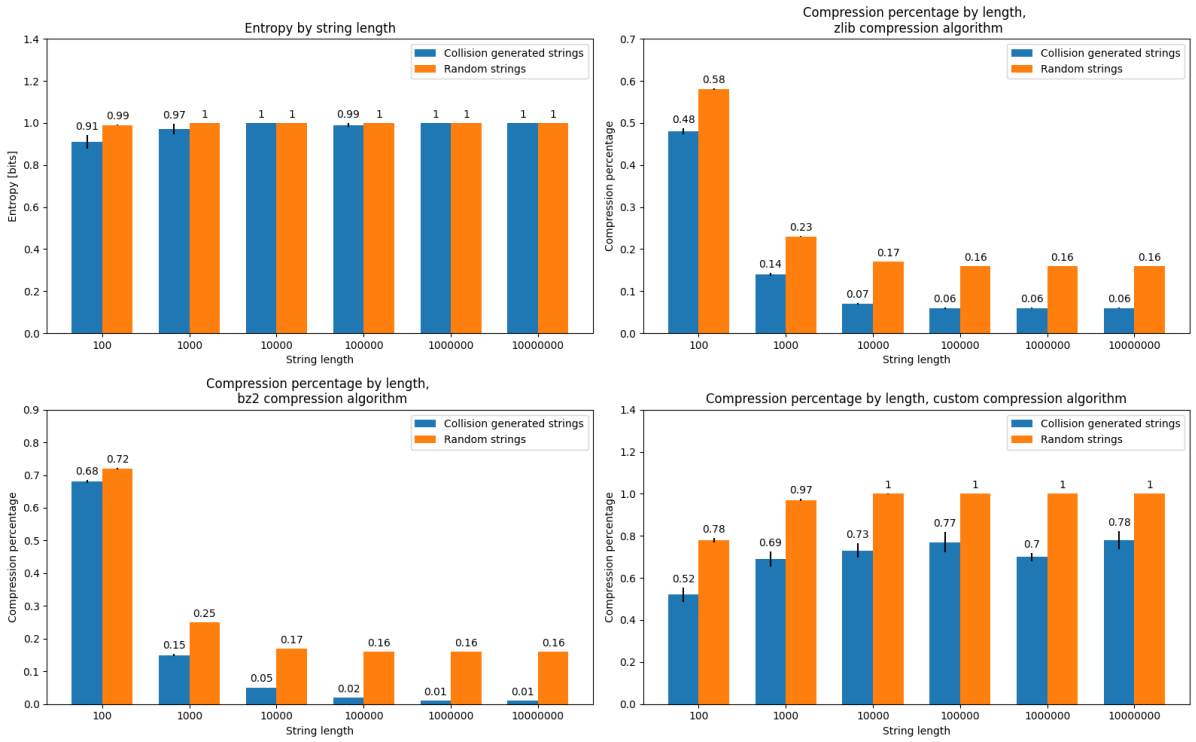

Figure 3. Performance of the chosen algorithms on both collision generated strings and random generated (RNG) strings. Top left is the entropy comparison, top right is the zlib algorithm compression percentage, bottom left is the bz2 algorithm compression percentage, and bottom right is the custom compression algorithm compression percentage (example compression: $aabccc \rightarrow 2ab3c$). Depicted results were averaged over 40 runs of randomized initial starting conditions (random initial velocity, position and masses of the three particles). Random strings perform substantially better than collision generated strings, especially with the bz2 compression algorithm, where the collision generated strings get compressed to almost 1% of the original length.

but slightly different results. Furthermore, due to floating point errors, for any small change in initial conditions there can be a bigger change in the final result due to the accumulation of rounding errors. This is why there is no apparent intrinsic structure in the resulting distribution of Z scores and the resulting small changes in mass ratios can produce vastly different results.

It is interesting to compare these results with the results of other compression algorithms, Figure 6. The custom compression algorithm produces the same results as the Z score of the runs test does. A compression percentage of 1 is obtained in a similar area of the mass ratios. However, the bz2 algorithm does not conform to this result. The compression percentage is way lower than the one found in Figure 3 for random strings. It does not increase anywhere in the mass ratio space and only gets smaller in certain areas. The zlib algorithm probably exploits correlations within the string generation mechanism, correlations which runs test cannot detect. What is confusing is that the areas in which the custom algorithm performs relatively better are the areas in which the bz2 algorithm performs relatively worse, and vice versa. We have no explanation for why this is happening because that would require in-depth understanding of the algorithms.

While none of this is definite proof of the ergodicity of the system it does tell us which mass ratios are relatively better at exploring the phase space of the whole system.

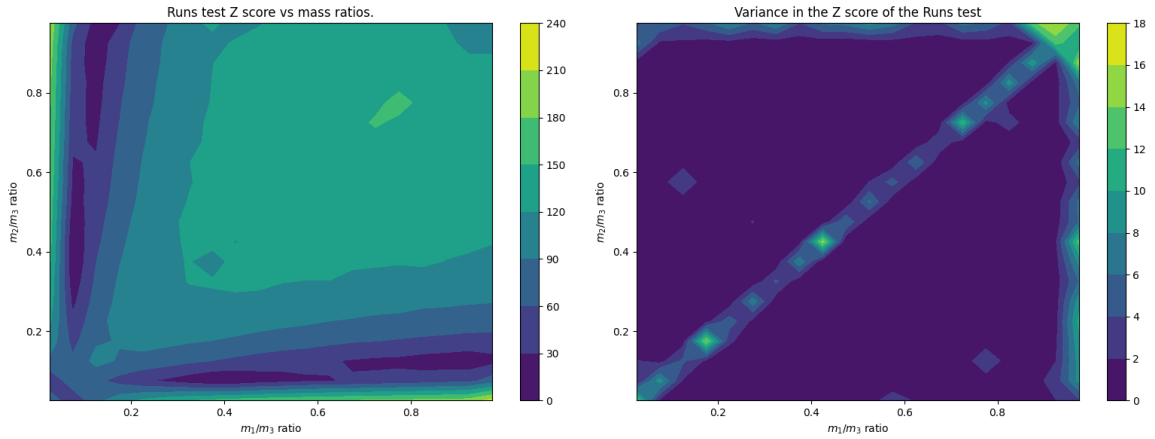

Figure 4. Results of the Wald-Wolfowitz runs test on collision generated strings with varying mass ratios. For each combination of mass ratios, ten strings were produced from randomized initial positions and velocities, the results were averaged and shown in the figure. On the left, we see that most of the input space we are working on produces relatively high Z scores. There are a few valleys of low Z scores. Variance in the Z score is reasonably low, barring few isolated areas, enforcing the belief that mass ratios 'control' the randomness of the generated strings.

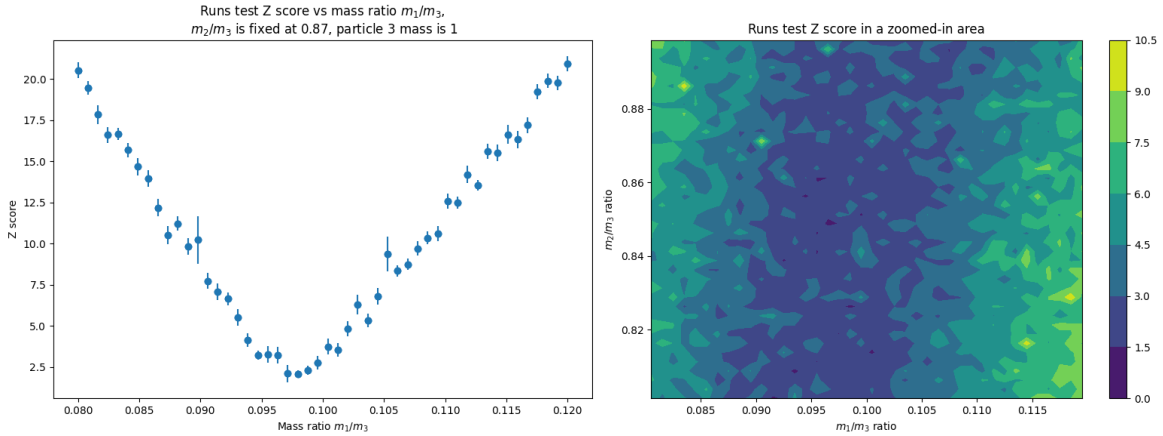

Figure 5. Z scores through an m_2/m_3 cut of the mass ratios space are depicted on the left. Minimum value of $Z = 2.08$ occurs for $m_1/m_3 = 0.097$. Although not small enough to confirm statistical randomness with 5% confidence, it is reasonable to assume that the surrounding area might contain some such mass ratios. Data were sampled from 50 mass ratios; 40 experiments with randomized initial conditions were conducted for each mass ratio. A zoomed in area of the runs test Z score for differing m_1/m_3 and m_2/m_3 ratios is depicted on the right. m_1/m_3 ratio runs from 0.08 to 0.12 while m_2/m_3 ratio goes from 0.8 to 0.9. For each combination of mass ratios, ten experiments were conducted with randomized initial positions and velocities. No intrinsic structure in the distribution of the scores is readily visible.

4 Conclusion

In this project, we came up with a novel way of investigating how do three particles on a ring explore their phase space. The method involves the creation of binary '01' strings depending on the particles' collision history - after a specific collision, only two things can happen: either one particle collides with the third one, or the other one does. These collision generated strings were compared to the Random Number Generator strings. This comparison is important because it gives us a 'benchmark' against which we can test all our results.

We started with the inspection of information theory entropy and the application of various compression algorithms on differing string lengths. It was found that entropy approaches the value of 1 bit early on and does not change with increasing string lengths.

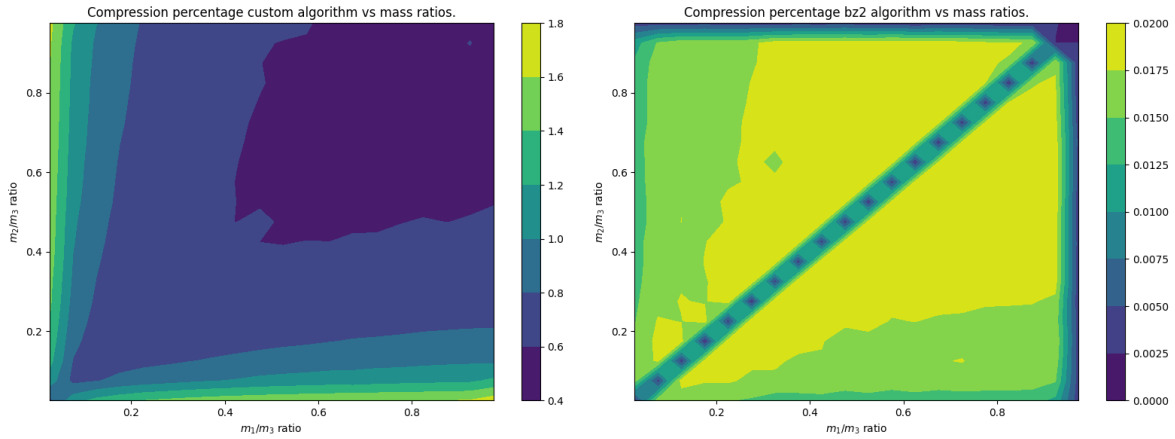

Figure 6. Custom compression algorithm and bz2 algorithm compression results in the plane of mass ratios. The custom compression algorithm reproduces results similar to the runs test, while the bz2 algorithm does not. Bz2 algorithm compresses the strings much better regardless of the mass ratios, and the results do not look similar to the results obtained for RNG strings. Further investigations are required.

Similarly, compression percentages stabilize with increasing string lengths and already at length of 10000, compression percentages start to stabilize. It was found that RNG strings do not get as compressed as collision generated strings do, and their compression limit is around 16%. On the other hand, collision generated strings get compressed to percentages as small as 6% with the zlib compression algorithm and as small as 1% of the original size with the bz2 compression algorithm. Since the entropy of both collision generated strings and RNG strings are 1 bit, the non-randomness of the collision generated strings must come from the distribution of the digits within the string and not because of the unequal amounts of digits in the string.

To further explore the system, we utilized the Wald-Wolfowitz runs test and tested the randomness of collision generated strings for different masses of the particles. It was found that mass ratios $\frac{m_1}{m_3} \in [0.1, 0.2]$ and $\frac{m_2}{m_3} \in [0.4, 1]$ (or vice versa, the results are symmetric around the $\frac{m_2}{m_3} = \frac{m_1}{m_3}$ line) produce the lowest runs test scores and are the closest approach to randomness. Further exploration of the area of interest produces some strings that pass the runs test with 95% confidence. Unfortunately, the bz2 and zlib compression algorithms do not agree with these results. Their compression ratios of collision generated strings are lower than for RNG strings for any choice of mass ratios. This is something that needs further investigation; insights into the algorithm's inner workings would surely help.

Careful inspection of the system made rise to some ideas on how to infer the probability of certain types of collision from the inspected momentum probability distribution function. These ideas are outlined in the Appendix B. The whole procedure requires further work. However, it could be a way of obtaining mass ratios for which the generated strings approximate the binomial distribution. These strings should pass the runs test and be the ones that best approximate truly random strings.

References

[1] Peter Vranas. "Epsilon-Ergodicity and the Success of Equilibrium Statistical Mechanics". In: *Philosophy of Science* 65.4 (1998), pp. 688–708. DOI: 10.1086/392667.

[2] Zhigang Zheng, Gang Hu, and Juyuan Zhang. "Ergodicity in hard-ball systems and Boltzmann's entropy". In: *Phys. Rev. E* 53 (4 Apr. 1996), pp. 3246–3252. DOI: 10.1103/PhysRevE.53.3246. URL: https://link.aps.org/doi/10.1103/PhysRevE.53.3246.

[3] Nándor Simányi. "Proof of the Ergodic Hypothesis for Typical Hard Ball Systems". In: *Annales Henri Poincaré* 5.2 (Apr. 2004), pp. 203–233. ISSN: 1424-0661. DOI: 10.1007/s00023-004-0166-8. URL: https://doi.org/10.1007/s00023-004-0166-8.

[4] Graeme J. Ackland. "Equipartition and ergodicity in closed one-dimensional systems of hard spheres with different masses". In: *Phys. Rev. E* 47 (5 May 1993), pp. 3268–3275. DOI: 10.1103/PhysRevE.47.3268. URL: https://link.aps.org/doi/10.1103/PhysRevE.47.3268.

[5] Haihong Li, Zhoujian Cao, and Gang Hu. "Analytical solution of space probability distributions of particles in a one-dimensional ring". In: *Phys Rev E Stat Nonlin Soft Matter Phys* 67.4 Pt 1 (Apr. 2003), p. 041102. DOI: 10.1103/PhysRevE.67.041102.

[6] S. G. Cox and G. J. Ackland. "How Efficiently Do Three Pointlike Particles Sample Phase Space?" In: *Phys. Rev. Lett.* 84 (11 Mar. 2000), pp. 2362–2365. DOI: 10.1103/PhysRevLett.84.2362. URL: https://link.aps.org/doi/10.1103/PhysRevLett.84.2362.

[7] Tahir Çagin and John R. Ray. "Fundamental treatment of molecular-dynamics ensembles". In: *Phys. Rev. A* 37 (1 Jan. 1988), pp. 247–251. DOI: 10.1103/PhysRevA.37.247. URL: https://link.aps.org/doi/10.1103/PhysRevA.37.247.

[8] Sheldon Lee Glashow and Laurence Mittag. "Three rods on a ring and the triangular billiard". In: *Journal of Statistical Physics* 87.3 (May 1997), pp. 937–941. ISSN: 1572-9613. DOI: 10.1007/BF02181254. URL: https://doi.org/10.1007/BF02181254.

[9] Ronald L. Graham, Steven Kay Butler, and National Science Foundation (U.S.) *Rudiments of Ramsey theory*. American Mathematical Society, 2015. ISBN: 9780821841563.

[10] Stephen Wolfram. *Note (a) for Defining the Notion of Randomness: A New Kind of Science — Online by Stephen Wolfram [Page 1068]*. URL: https://www.wolframscience.com/nks/notes-10-3--inevitable-regularities-and-ramsey-theory/.

[11] M. G. Kendall and B. Babington Smith. "Randomness and Random Sampling Numbers". In: *Journal of the Royal Statistical Society* 101.1 (1938), pp. 147–166. ISSN: 09528385. URL: http://www.jstor.org/stable/2980655.

[12] George Marsaglia. *The Marsaglia Random Number CDROM including the Diehard Battery of Tests*. 1995. URL: https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/.

[13] James Bellamy. "Randomness of D Sequences via Diehard Testing". In: *CoRR* abs/1312.3618 (2013). arXiv: 1312.3618. URL: http://arxiv.org/abs/1312.3618.

[14] NIST agency of U.S. Department of Commerce. *Runs Test for Detecting Non-randomness*. 2003. DOI: https://doi.org/10.18434/M32189. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35d.htm.

[15] Mark Adler and Jean-loup Gailly. *zlib*. 1995. URL: https://github.com/madler/zlib.

[16] Julian Seward. *bzip2*. 1996. URL: https://gitlab.com/bzip2/bzip2/.

A Code

All of the code, scripts and plots used in the project can be found in this GitHub repository.

B Future research: inferred analytical probability of certain types of collisions

All possible collisions in this system can be classified into four groups. Depending on the collision type, a series of consecutive identical generated digits might be broken (e.g. 0000... will be generated until a specific collision occurs). The types are as follows:

- Initial momenta of two particles differ in sign; two particles collide head on one into the other (the best analogy would be a head on collision between two cars). There are two outcomes of interest for this type of collision:
  1. Particles reverse their directions and continue travelling in separate directions. Usually, this type of collision does not break a consecutive digit streak.
  2. Both particles travel in the same direction post collision. This type of collision usually breaks a consecutive digit streak.
- Initial momenta of two particles have the same sign; two particles collide, with one particle 'catching up' to the other one (the best analogy would be two cars on a highway, one crashing into the other from behind). The two outcomes are:
  1. The particle that catches up to the other particle reverses the direction of its traversal, and the other particle gets a velocity boost. This type of collision leads to the creation of identical digits.
  2. The particle that catches up to the other particle slows down but does not reverse its direction; the other particle gets a velocity boost. This type of collision leads to the creation of different digits and the breaking of a digit streak.

With the momentum distributions in mind, it should be possible to find mass ratios that give equal probabilities of identical consecutive digits happening and differing consecutive digits happening. Phrased differently, there should be mass ratios that, after a collision, give probability $\frac{1}{2}$ of the same digit being generated in the next collision and a probability of $\frac{1}{2}$ of a differing digit being generated in the following collision. Of course, this still does not mean that the produced digit string is intrinsically random as it is produced via

a perfectly deterministic and correlated system. However, the generated string should resemble an RNG string of a binomial distribution. This string should be the one that sees the least compression when run through the compression algorithms and should resemble the compression results of RNG strings. The case of a head on collision with post collision velocities pointing in the same direction is considered below.

The system is under the following constraints

$$p_1 + p_2 + p_3 = 0$$

and

$$\frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2} + \frac{p_3^2}{2m_3} = E_0.$$

The probability density distribution is given by

$$P(p_i) = \frac{1}{\pi}(p_{i,max}^2 - p_i^2)^{-1/2}$$

where $p_{i,max}$ is the maximum momentum particle i can take on.

Start by considering the probability that particle 1 collides in a way such that both particles travel in the same direction after the collision.

$p_{1,max}$ can be found to be (using the method of Lagrange multipliers; we are minimizing the function $\frac{p_2^2}{2m_2}+\frac{p_3^2}{2m_3}$ with the constraint $p_{1,max}+p_2+p_3 = 0$; results are $p_2 = \frac{p_{1,max}}{1+\frac{m_3}{m_2}}, p_3 = -\frac{p_{1,max}}{1+\frac{m_2}{m_3}}$):

$$E_0 = \frac{1}{2}\left(\frac{(m_2+m_3)^2 + m_1m_2 + m_1m_3}{m_1(m_2+m_3)^2}\right)p_{1,max}^2$$

and we will use this in the final probability calculation.

Now looking at a collision and assuming the initial momenta are $p_1 > 0$ and $p_2 < 0$:

$$q_1 = \frac{m_1 - m_2}{m_1 + m_2}p_1 + \frac{2m_1}{m_1 + m_2}p_2.$$

To break the 'chain' of consecutive digits in a row, we need that after the collision $q_1 > 0$ and is travelling in the same direction as the other recoiled particle. This requirement is satisfied for $q_1 > 0$, and doing some rearranging leads to

$$p_1\left(\frac{-m_1 - m_2}{m_1 - m_2}\right) > \frac{2m_1}{m_1 - m_2}p_3.$$

Start with considering the $m_1 > m_2$ case. Rearranging leads to

$$p_1 < \frac{2m_1}{m_1 + m_2}p_3.$$

p_3 can be eliminated from the inequality using the conservation of momentum $p_2 = -p_1 - p_2$ and the conservation of energy to give the inequality only in terms of p_1. The result is:

$$p_1^2\left(1 + \frac{m_1m_3(3m_1m_3 + 4m_2m_3 + 4m_2^2 + 4m_1m_2)}{(m_1m_2 + 2m_1m_3 + m_2^2 + m_2m_3)^2}\right) \tag{3}$$

$$< \frac{2m_1^2m_2m_3(m_2+m_3)}{(m_1m_2 + 2m_1m_3 + m_2^2 + m_2m_3)^2}E_0. \tag{4}$$

Now, we can write out E_0 in terms of $p_{1,max}$ and the particle's masses, and the introduction of the constants $\alpha = \frac{m_1}{m_3}$ and $\beta = \frac{m_2}{m_3}$ leads to

$$p_1^2 < \frac{\alpha\beta[(\beta+1)^2 + \alpha\beta + \alpha]}{(\beta+1)[(\alpha\beta + 2\alpha + \beta^2 + \beta)^2 + \alpha(3\alpha + 4\beta + 4\beta^2 + 4\alpha\beta)]} p_{1,max}^2.$$

The original momentum probability distribution function can be transformed into the momentum squared probability density function:

$$P(a^2 \leq y < b^2) = \frac{1}{2\pi} \int_{a^2}^{b^2} \frac{1}{\sqrt{y(p_{1,max}^2 - y)}} dy$$

$$P(a^2 \leq y < b^2) = -\frac{1}{2\pi} \left(\arcsin\left(\frac{p_{1,max}^2 - 2y}{p_{1,max}^2} \right) \right) \Bigg|_{a^2}^{b^2}$$

where $y = p_1^2$. Finally, we can find

$$P(p_1^2 < K p_{1,max}^2) = \frac{1}{4} - \frac{1}{2\pi} \arcsin(1 - 2K).$$

Similarly, had it been assumed that $m_2 > m_1$ the equation we get is

$$P(p_1^2 > K p_{1,max}^2) = \frac{1}{4} + \frac{1}{2\pi} \arcsin(1 - 2K).$$

Plotting this resulting probability for differing mass ratios results in Figure 7. The results do not seem to agree with what has been found for the Runs test. However, this could be due to only the probability of particle 1 being considered, and that other types of collisions should be considered as well. Hopefully, this sample calculation shows that it should be possible to calculate the probability of the runs test results on the generated strings analytically. The complete calculation is outside the scope of this project.

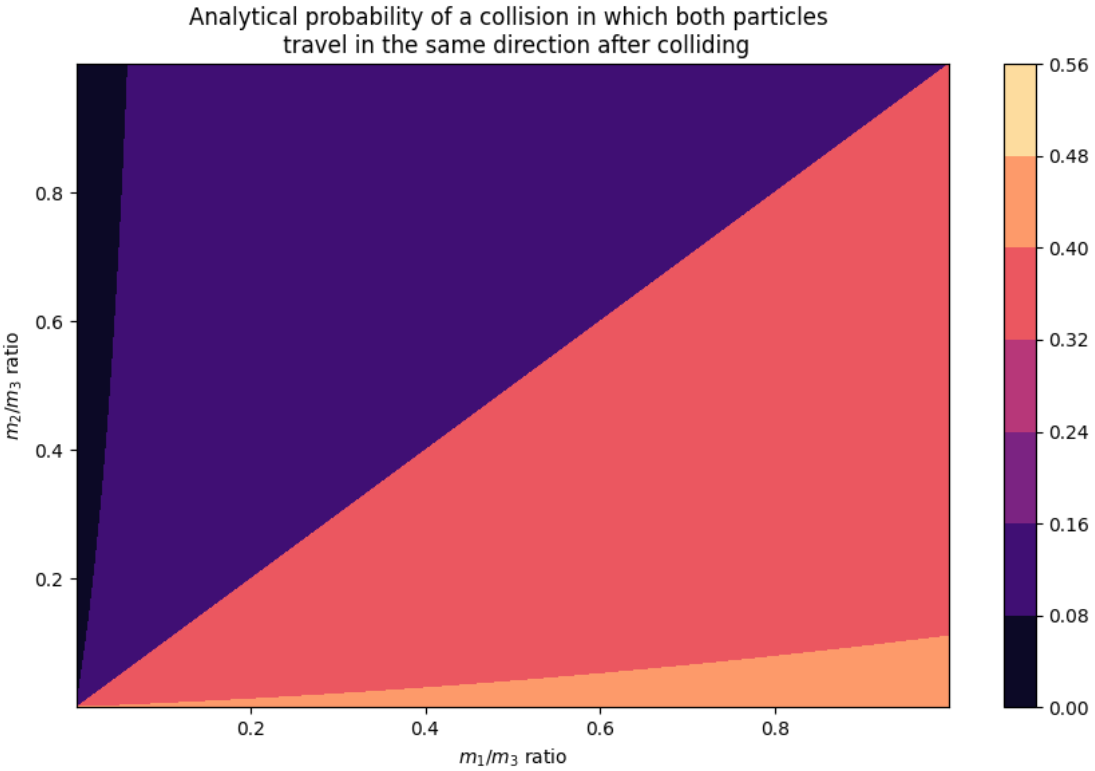

Figure 7. Probability of both particle 1 and particle 2 travelling in the same direction after a head-on collision for differing mass ratios. The plot is not symmetric because it had been assumed that the particle 1 and 2 collide, with both particles travelling to the right after the collision. Addition of further calculations, such as that both particles can travel to the left after the collision should make the plot symmetric.