Developing Dynamic Toys: Investigating a Biased Galton Board and the Brachistochrone problem

Eoghan Phelan

School of Mechanical and Materials Engineering

MEEN 30120 BE Mechanical Engineering Project

Final Report

Supervisors: Dr. Vikram Pakrashi, Dr. Kevin Nolan

21st of April 2020

UCD School of Mechanical and Materials Engineering
Report Submission Form

This form should be completed and signed. It should be incorporated into your submission (PDF and hard copy versions) and should appear as a single page immediately after the title page.

Student Name: Eoghan Phelan

Student Number: 15430698

Report Title: Developing Dynamic Toys; Investigating a Biased Galton Board and Brachistochrone models

Plagiarism

Plagiarism is a serious academic offence and is comprehensively dealt with on UCD's Registry website (https://www.ucd.ie/governance/resources/policypage-plagiarismpolicy). It is a student's responsibility to be familiar with the University's policy on plagiarism. All students are encouraged, if in doubt, to seek guidance from an academic member of staff on this issue. The UCD policy document on plagiarism states that "the University understands plagiarism to be the inclusion of another person's writings or ideas or works, in any formally presented work (including essays, theses, projects, laboratory reports, examinations, oral, poster or slide presentations) which form part of the assessment requirements for a module or programme of study, without due acknowledgement either wholly or in part of the original source of the material through appropriate citation. Plagiarism is a form of academic dishonesty, where ideas are presented falsely, either implicitly or explicitly, as being the original work of the author. While plagiarism may be easy to commit unintentionally, it is defined by the act not the intention. The University advocates a developmental approach to plagiarism and encourages students to adopt good academic practice by maintaining academic integrity in the presentation of all academic work".

Declaration of Authorship

I declare that all material in this submission is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

Signature: *Eoghan Phelan* Date: **20/4/20**

Abstract:

Both the Galton Board and Brachistochrone curve can be classified as dynamical systems. The Galton board has been used for over a half-century as a mechanical tool to illustrate the formation of normal distributions as well as the Central Limit Theorem. Here, it was used as the main instrument to show what occurs when the original system is altered, and bias is introduced. Utilizing a constructed python script in conjunction with the 3D software Blender, simulations were conducted on numerous different Galton board designs. Initial findings showed that varying scatter peg diameter could control the width of the Gaussian curve's standard deviation. Further developments drafting the worked script allowed a controlled bias for the left or right to be added within the scatter peg configuration. This addition gave simulation results that resembled a log-normal distribution opposed to a Gaussian form. Evaluating these outcomes with the addition of MATLAB allowed with a degree of significance that the particles did indeed possess multiple similarities associated with a log-normal distribution. Final efforts exploring the Galton Board showed that a designated randomised peg variation for the system seemed to revert to the original normal distribution seen initially.

Efforts made investigating the Brachistochrone problem centred around employing Blender to highlight its mechanical properties. At the end of proceedings, another developed script was drafted that could allow virtual simulations to be conducted within seconds. The goal of this work as well that done on the Galton Board was to shed light on aspects of Digital learning and its importance in an evolving pedagogical landscape. In doing so it was indented to increase the overall use of digital teaching tools across UCD in the hopes it could be an exemplar to other universities and institutions.

Table of Contents

Chapter **Page**

1 INTRODUCTION

1.1 Introduction…… 1
1.2 Research Hypothesis…… 2
1.3 Thesis aims…… 3
1.4 Methodology/Approach…… 4
1.5 Thesis outline…… 4

2 TECHNICAL FOUNDATIONS and LITERATURE REVIEW

2.1 Historical, Scientific and Technical Background
2.1.1 Galton Board…… 5
2.1.2 Brachistochrone model…… 8
2.2 Background into Blender and free body simulations…… 14
2.3 Context linking the project and Digital Learning……15

3 OVERALL METHODOLOGY

3.1 Galton Board investigation…… 16
3.2 Brachistochrone model…… 32

4 Discussion of Results

4.1 Discussion, Conclusion and Suggestions for possible future research…… 41

References…… 44

List of Figures

Page

Fig 2.1.1 - Galton's 1889 illustration of the quincunx…….. *6*

Fig 2.1.2 - Binominal probability representation of each falling particle……... *6*

Fig 2.1.3- Binominal coefficients imprinted on the array of scatter pegs…….. *8*

Fig 2.2.1- Galileo's solution…….. *9*

Fig 2.2.2- Light ray passing through a series of media with different refractive index…… *9*

Fig 2.2.3- Trace of a cycloid, solution to the brachistochrone problem……..... *12*

Fig 2.3.1- User interface for Blender 3D software…….…….... *14*

Fig 3.1.1- Initial virtual Galton board configuration……..… *16*

Fig 3.1.2- Final distribution of the initial square particle simulation, with an imprinted similar gaussian representation…… *16*

Fig 3.1.3- Galton board configuration example A……...……..… *17*

Fig 3.1.4- Galton board configuration example B……...……...… *17*

Fig 3.1.5– Example flow chart representation (n = 12)……..…… *19*

Fig 3.1.6- Gaussian distribution of initial scripted simulation…….…….. *20*

Fig 3.1.7- Distribution with narrower standard deviation……..…... *20*

Fig 3.1.8- Distribution with wider standard deviation……..... *20*

Fig 3.1.9- Skewed distribution to the right…….……...….. *21*

Fig 3.1.11- Skewed distribution to the left…….……..…..... *21*

Fig 3.1.12- Lognormal distribution graph…….. *22*

MATLAB Fig 1.1- Normal Dist. cftool fit with accompanying peg configuration..... *23*

MATLAB Fig 1.2- Log-normal Dist. cftool fit with accompanying peg configuration..... *24*

MATLAB Fig 1.3- Log-normal fitted probability density function..... *25*

Fig 3.1.13- Traced path of randomly selected particles *26*

MATLAB Fig 1.4- PDF comparison for increasing bias parameter..... *27*

MATLAB Fig 1.5- Log normal probability plot vs distribution data..... *27*

MATLAB Fig 1.6- PDF & Log normal probability plot for increased rows (n =16)..... *28*

Fig 3.1.14- Created array of random scatter pegs..... *30*

MATLAB Fig 1.7- Distribution comparison for simulations of a random peg variation..... *30*

MATLAB Fig 1.8- CDF comparison for simulations of a random peg variation..... *31*

Fig 3.2.1- Developed brachistochrone mesh..... *34*

Fig 3.2.2- Test of the Brachistochrone and two other alternative paths..... *34*

Fig 3.2.3- Conducted simulation first and last frame comparison *35*

Fig 3.2.4- Tautochrone simulation frame analysis..... *36*

MATLAB Fig 2.1- Plotted data of displacement vs time..... *38*

MATLAB Fig 2.2- Calculated velocity data of vs time..... *38*

MATLAB Fig 2.3- Direction of velocity vector vs time..... *39*

MATLAB Fig 2.4- Displacement in x & overall velocity vs time..... *39*

MATLAB Fig 2.5- Tautochrone simulation, direction of velocity vector vs time..... *40*

ACKNOWLEDGMENTS

I would like to acknowledge everyone who played a role in my academics at my time at UCD. First, my parents, who supported me all the way through. Secondly, each member of staff and lecturer in the school of Engineering who has guided my learning journey over the past 5 years. And lastly, I want to give massive appreciation to both my supervisors, Dr. Vikram Pakrashi and Dr. Kevin Nolan. Each provided patient advice and guidance throughout this research process and at several moments I would have struggled without them. Thank you again for all for your unwavering support.

CHAPTER 1

1.1 Introduction:

Dynamical toys allow one to see first-hand the variety of concepts of physics and other mathematical principles in action. They are an easy and inexpensive way to visualize certain theories that would otherwise be difficult to fully grasp. the laid-out title and further description of this given project were set out as follows.

Title: "*Developing Dynamic Toys: Investigating a Biased Galton Board and the Brachistochrone problem*".

Description: "*Conception, development, simulation, testing, and validation of complex behaviour of dynamical educational systems and models*."

The first dynamic toy being investigated being as stated is a *Galton Board*, or also referred to as a *quincunx* [1]. This device is a statistical model invented by Sir Francis Galton to demonstrate the central limit theorem. More specifically, to show with a large sample size a binomial distribution approximates to a normal distribution. The model demonstrates how order can arise from what seems to be chaotic behaviour. In conjunction with this project, experimental simulations are to be conducted on a virtual Galton board by utilizing the powerful computer application known as Blender. In turn, this will lead to a greater understanding of the principles and mechanics of the device. Another key component within the title is the term *biased*. Investigating ways of which to integrate bias into a Galton board design can be seen to be large the focus for the first portion of this project developing these dynamic toys. Once significant progress has been reached with exploring the Galton board, the succeeding efforts will be focused on another interesting chaotic toy model known as the Brachistochrone curve [2]. This curve represents the fastest route of descent that can be taken between point A and a lower point B. The Problem itself has astonished mathematicians for centuries, most notably associated with Johann Bernoulli in the late 1600s. By once again

implementing the use of Blender, a greater understanding of the dynamic system can hopefully be achieved.

1.2 Research Hypothesis:

Leading into this research project it was essential to look at it in its two separate parts. The initial hypothesis is drawn with regards to investigating the Galton board as to what extent can the final distribution be controlled? These outcomes and observations are solely dependent upon the formation and pattern of the scatter pegs within the designed system. After continued experiments and simulations it will be apparent if a non-uniform pattern of pegs can consistently produce a predictable distribution for a large number of particles. This goes for a controlled bias and complete random scatter peg variation. The objective is to find significant results to adopt this hypothesis. If significant results do not support this hypothesis, it can be said that only uniform pegs can generate a predicted spread of particles. This being a normal distribution for the original Galton board design.

Efforts made on the Brachistochrone problem centre around making an adaptive software that showcases the extent of the model's properties. In achieving this it is shout out to derive and reinforce existing mathematical identities that are associated. In conjunction with the roots of this project with digital learning, it can be thought to what degree can this complex system be made easy to grasp for users while providing a powerful visual representation and analysis. This can be said for both the Brachistochrone and Galton board. Combined, both aspects set to provide a significant impact on the overall pedagogy of mechanics specifically for the school of mechanical and materials Engineering here at UCD.

1.3 Thesis Aims:

To continually be aware the project is making significant progress as the time went by, it was important to set out concrete objectives at the beginning stages of the planning process. The key objectives to be adhered to concerning the Galton board were as follows.

1. Develop demonstrative tools for a digitally-driven method of teaching mechanics within UCD
2. Investigate the possibility of controlling distributions from a Galton board by modifying its design.
3. Develop a python script that generates these Galton board configurations within Blender.
4. Evaluate the implementation of bias to the conducted simulations.

Similarly, the Brachistochrone problem had a set of objectives that were set out to be hopefully achieved.

1. Implement additional scripting to further develop a digitally automated simulation to test the dynamic system.
2. Incorporate a simplistic interface for operating the code which would allow potential users to make avail of the finished product,
3. Acquire post-simulation analysis implementing available tools such as MATLAB or other software.

Keeping to this projected plan was a priority. However, if obstacles prevented progress it was important to adapt and change strategy accordingly.

1.4 Methodology:

The implementation of Blenders software was the main non-invasive method used to determine certain scenarios and test the research hypothesis. The decision to use such techniques is in line with the aims of this project was the overall flexibility and the powerful capability's that Blender provides. Using a physical approach to test configurations for both the Galton board and the Brachistochrone would be time-consuming and an expensive process. The qualitative experimental simulation techniques which will be used fits directly inline what the project brief outlines. With regards to conducting post-analysis on subsequent developments that are achieved, it was intended to incorporate MATLAB in conjunction with Blender. This allows a level of detail that would be not available solely within Blender as for the python libraries it contains not have sufficient facilities.

1.5 Thesis Outline:

The remainder of this report will be structured into two sections. The first chapter investigates the literature findings on all aspects regarding this research project. Both the Galton board and The Brachistochrone models have unique and interesting beginnings that will be explored in greater detail. Once that is concluded the mathematical and technical aspects of each will be touched on and explained fully. Linking these principles within the confines of this project and showing the methods in which, these concepts are simulated using the capacity of Blenders physics engine to do so is an important section that is addressed. Lastly, sceptics will question what the purpose of these simulations of multiple dynamic systems. It is with that by providing key evidence on ways in which the findings of the project can aid specific forms of educational methods was a priority.

The following section details the evolution of this project from the beginning up until completion. For both project aspects, the specific methods that were involved are covered extensively. The final chapter lays out and unravels key results gathered throughout the process while also providing potential avenues for continued research and work in the future.

CHAPTER 2

Literature Review:

2.1.1 Historical, Scientific and Technical Background: Galton Board.

The inventor of the Galton board Sir Francis Galton was born on the 16th of February 1822, in the United Kingdom. His grandfather Erasmus Darwin was also the grandfather of one Charles Darwin, the man that originated the theory of evolution. From an early age, Galton showed promising skills in mathematics but originally went to university to study medicine. After becoming frustrated with studying medicine at first, Galton went to Cambridge University instead to study mathematics. He also later went back to finish his degree in medicine. When Galton's father died early in his adult life, he was left with a quite large inheritance. These funds freed him from having to work for the next six or so years. Within that time, he travelled extensively across Europe, Africa, and Asia. When he returned home in 1850 at the age of 28, he was elected to the Royal Society of England [3].

In this next chapter of Galton's life, he began looking into the field of Eugenics. It is defined as the practice or advocacy of improving the human species by selectively mating people with specific desirable hereditary traits. Although the field of Eugenics has since been widely disproven, at the time Galton believed that certain physical characteristics such as weight and height and other psychological characteristics could be inherited and improved upon with successive generations of selective breeding. After running studies based on traits father and sons share, Galton concluded that the traits of successive generations returned to a mean value. In simpler terms, the offspring of short parents will be slightly taller, and the children of taller parents will tend to be slightly shorter. This is known as regression to the mean.

During Galton's time studying eugenics, he devised instruments to aid in the process of data collection and illustration of these methods. The most well-known of these inventions was the Galton board. The device was used to illustrate the laws of error and hereditary processes.

This Galton board, also referred to as a *quincunx*, was a tin rectangular box with a glass front. Several horizontal pins were embedded through the back of the board like nails. These numerous pins were arranged in a regular triangular array or quincunx pattern. This arrangement consists of five objects with four located at the corners of a square or rectangle and the fifth at its centre, commonly seen for the five on a typical dice. Micro led pellets were dropped into the board with a funnel and were collected in evenly distributed bins/slots at the bottom of the board. Galton observed the formation of the pellets at the bottom formed a normal distribution curve. After conducting many more sophisticated experiments with this *Galton Board*, he proved that normal distribution was typically a sum of many binominal distributions. This hypothesis is essentially the Central Limit Theorem that is used today in the field of probability and statistics [3].

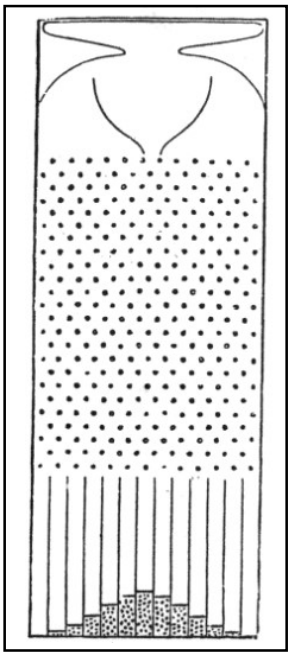


Figure 2.1.1- Galton's 1889 illustration of the quincunx [4]

As mentioned above, the central limit theorem is a statistical concept that establishes that the distribution mean of the random samples approaches a normal distribution as the sample size increases. This applies to the Galton board as each particle has a probability function that follows a binomial distribution. If a particle bounces to the right *k* times on decent (and to the left on the remaining pegs) it will then settle in the kth bin counting from the left. Denoting the number of rows of pegs in a Galton Board by n, the number of paths to the kth bin on the bottom is given by the binomial coefficient $\binom{n}{k}$.

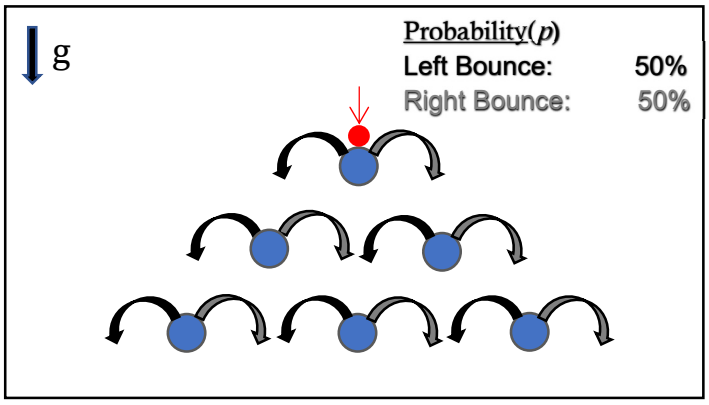


Figure 2.1.2- Binominal probability representation of each falling particle

With the chances of a particle bouncing right and left on a peg being nearly 50/50 the way the Galton board is designed, the probability function that the ball ends up in the k^{nt} bin is given by the following [5];

$$\binom{n}{k} p^k (1-p)^{n-k} \qquad \text{Eq 2.1.1}$$

This probability function for every single particle as mentioned above follows that which is a *binomial distribution*. As each particle representing an independent Bernoulli trial, over many instances (i.e. large number of particles and pegs), the central limit theorem states that the overall probability mass function can be approximated as a *normal distribution* [6].

With each having a probability *p* of success (a binomial distribution with *n* trials), converges to the probability density function of the normal distribution. The normal distribution is seen to have a mean and standard divination as the following [5].

$$\mu_x = np \qquad \text{Eq 2.1.2}$$

$$\sigma_x = \sqrt{np(1-p)} \qquad \text{Eq.2.1.3}$$

As *n* grows large, for *k* and probabilities for a right bounce *p* and left *q*, it gives that the function approximates to a gaussian distribution [5].

$$\binom{n}{k} p^k q^{n-k} \approx \frac{1}{\sqrt{2\pi npq}} e^{-\frac{(k-np)^2}{2npq}}, \qquad p+q=1, p, q > 0 \qquad \text{Eq 2.1.4}$$

The Galton board is also synonymous with a commonly used mathematical principle known as "*Pascals Triangle"* [7]. The model is a triangular array of binomial coefficients. It is constructed by starting the first coefficient on the 0th row, donated by a binominal 1. Each subsequent row and coefficient below are achieved by summing the number above to the left with the number and above to the right of that position. Blank entries are denoted with a value of 0. Assigning each binomial coefficient with each continuous rows of pegs on the Galton board, each value represents the number of different paths a particle can take to reach that

specific peg. Therefore, it is expected that many of the particles will fall in the slots within the boards centre. And many fewer particles will be expected on the outskirts of the bins as this requires a considerable amount of a bounce in one particular direction rather than the other.

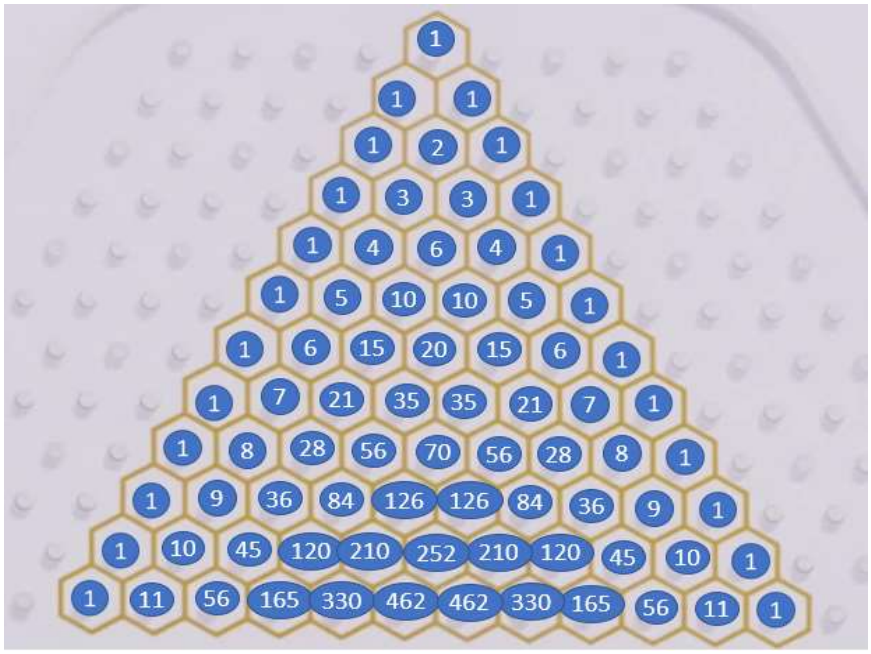


Figure 2.1.3- Binominal coefficients imprinted on the array of scatter pegs [4]

2.1.2 Historical, Scientific and Technical Background: Brachistochrone Problem.

In June of 1696, Johann Bernoulli set the coveted Brachistochrone problem in the Acta Eruditorum mathematics journal as he set a challenge to what he stated, “the most acute mathematicians of the entire world”. although knowing how to solve it himself, he challenged others to solve the problem as follows:

“*Given two points A and B in a vertical plane, what is the curve traced out by a point acted on only by gravity, which starts at A and reaches B in the shortest time?*” [2].

In Greek, ‘brachistos’ means ‘the shortest’ and ‘chronos’ means ‘time’, hence the name ‘brachistochrone’. Bernoulli was not the first to tackle this topic. Galileo had studied the problem well before this back in 1638. His workings concluded that the point would reach B more quickly if it travelled along the separate line segments AC followed by CB where C is a point on an arc of a circle.

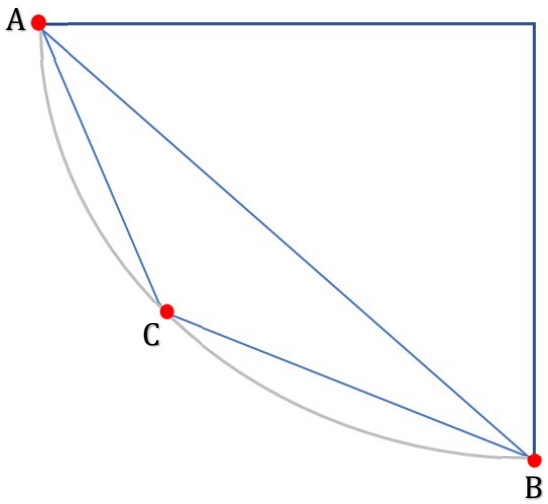


Figure 2.2.1- Galileo's solution

Although Galileo was correct in this thinking, it was an error when he next argued that the path of quickest descent from A to B would be an arc of a circle. When Bernoulli revisited and problem later in the century and posed the challenge to the world several different solutions were obtained. Perhaps the most notable coming from one Isaac Newton. According to Newton's niece, Isaac learned of the challenge at 4 pm and had solved it by 4 am the following morning. His solution was later committed to the Royal Society anonymously. Although Bernoulli is noted to have said he was aware that of Newton's work as he put it, "recognised the lion by its claw". Newton may have been the quickest to solve the problem, but the most elegant solution may have come from Bernoulli himself. His approach adopted an outlook on how light takes the least time from moving from one medium to another. He devised a system of a light ray passing through an infinite sequence of horizontal differentially thin optical media of varying refractive index with depth. As light travels from point A to a lower point B, its speed increases continuously as the increasing angle of refraction [2].

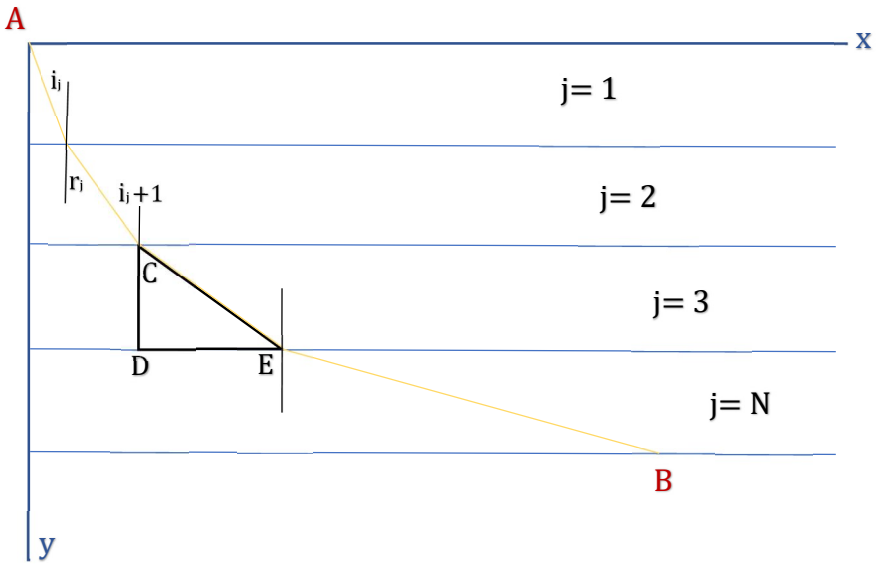


Figure 2.2.2- Light ray passing through a series of media with different values of refractive index.

Bernoulli recognised the similarity between the continuously increasing speed of a light ray and the increasing speed of a mass falling in a uniform gravitational field. Therefore, he concurred that by following a law similar to that followed by a refracting light ray, the falling mass must take minimum time. By substituting for the speed of light in Snell's law with the velocity v acquired by a falling mass in a uniform gravitational field. By applying the law of conservation of energy which states that the sum of potential energy and kinetic energy of a falling mass must remain constant. If the falling mass starts from rest, then the sum must be zero for each point along the path.

Considering figure 2.1.6 above, the j^{nt} layer has a refractive index of nj ($j = 1, 2, \dots N$). This gives a series of continuous refractions since refractive index changes continuously along the path of the ray. The angle of refraction of any of the intermediate refractions is equal to the angle of incidence for the next refraction.

$$\frac{\sin i_j}{v_j} = \frac{\sin r_{j+1}}{v_{j+1}} = k, \quad r_j = i_{j+1} \qquad \text{Eq 2.2.1}$$

The following can obtain an equation for the speed *v*, in terms of the height of decent using the law of conservation of energy.

$$\frac{1}{2}mv^2 + mgy = 0, or\ v = \sqrt{2gy} \qquad \text{Eq 2.2.2}$$

With the differential triangle CDE (containing the angle of refraction DCE) shown in ***Figure 2.1.6***, it can be shown that $(dy/ds) = sin\ r = kv = (v/a)$, where $k = (1/a)$ is the Bernoulli's constant, yielding the following.

$$CE = \sqrt{CD^2 + DE^2} \qquad \text{Eq 2.2.3}$$

$$ds = \sqrt{dx^2 + dy^2}, \quad or \quad dx = kv\sqrt{dx^2 + dy^2} \qquad \text{Eq 2.2.4}$$

$$dx = \frac{v\,dy}{\sqrt{a^2 - v^2}} \qquad \text{Eq 2.2.5}$$

Combining equations (2), (4) and (5) gives the following.

$$dx = \frac{\sqrt{ay}}{\sqrt{a^2-ay}}\ dy = \sqrt{\frac{y}{a-y}}\ dy = \frac{y}{\sqrt{ay-y^2}}\ \ dy \qquad \text{Eq 2.2.6}$$

Then adding and subtracting $\frac{a}{2\sqrt{ay-y^2}}\ \ dy$ to the RHS.

$$\sqrt{\frac{y}{a-y}}\ \ dy = \frac{a}{2\sqrt{ay-y^2}} - \frac{a-2y}{2\sqrt{ay-y^2}}\, dy \qquad \text{Eq 2.2.7}$$

Integrating across yields the bellow equation.

$$x = a\ \sin^{-1}\sqrt{\frac{y}{a}} - \sqrt{a^2-y^2} \qquad \text{Eq 2.2.8}$$

Bernoulli noticed this expression was that the same for the function of a cycloid. A cycloid is a path a point on the circumference of a circle takes when it rolls along a flat line. To prove it is indeed a cycloid, *y* can be replaced with the corresponding trigonometric function.

$$y = \tfrac{1}{2}a(1 - \cos\theta) \qquad \text{Eq 2.2.9}$$

Which gives.

$$x = a\sin^{-1}(\sin\tfrac{\theta}{2}) - a(\sin\tfrac{\theta}{2})\sqrt{\tfrac{1}{2}(1+\cos\theta)} \qquad \text{Eq 2.2.10}$$

$$x = \tfrac{1}{2}a\theta - \tfrac{1}{2}a\sin\theta \qquad \text{Eq 2.2.11}$$

Denoting a as the twice the radius of the circle that traces the cycloid $(\frac{a}{2} = r)$, the concluding expressions are achieved [8].

$$x = r(\theta - \sin\theta) \qquad \text{Eq 2.2.12}$$

$$y = r(1 - \cos\theta) \qquad \text{Eq 2.2.13}$$

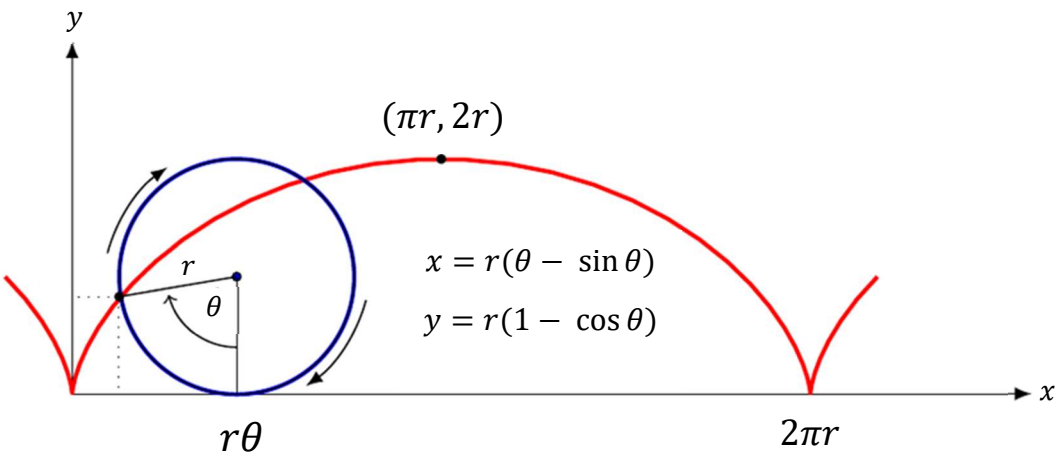


Figure 2.2.3- Trace of a cycloid, solution to the brachistochrone problem [9]

Upon concluding on his discovery Bernoulli is quoted to have said: “Furthermore, I think it is noteworthy that this identity is found only under the hypothesis of Galileo so that even from this we may conjecture that Nature wanted it to be thus”.

The Brachistochrone is also referred to the Tautochrone curve. This comes from the Greek prefixes *tauto-* meaning same, and *chronos* meaning time [10]. The curve has the property for which the time taken by a particle sliding the route is independent of the particle's starting point on the curve.

Returning to the previous *Eq.2.2.2* for the conservation of energy it can also be written as the following:

$$v = \frac{ds}{dt} = \sqrt{2gy} \qquad \text{Eq 2.2.14}$$

$$dt = \frac{ds}{\sqrt{2gy}} = \frac{\sqrt{dx^2+dy^2}}{\sqrt{2gy}} \qquad \text{Eq 2.2.15}$$

Thus, the time taken for the sphere to slide down the path can be given by:

$$t = \int \sqrt{\frac{dx^2+dy^2}{2gy}} \qquad \text{Eq 2.2.16}$$

Substituting the equations for a cycloid into the above integral yields:

$$t = \int_0^{\pi} \sqrt{\frac{2r^2(1-\cos\theta)}{2rg(1-\cos\theta)}}\, d\theta = \Big[\theta\sqrt{\frac{r}{g}}\Big]_0^{\pi} = \pi\sqrt{\frac{r}{g}} \qquad \text{Eq 2.2.17}$$

Now suppose the sphere is released at any secondary point (x_0,y_0). Taking Eq.2.14 and substituting gives:

$$\frac{ds}{dt} = \sqrt{2g(y - y_0)} \qquad \text{Eq 2.2.18}$$

Repeating the above steps, the time for a sphere to reach the bottom of the path yields:

$$t = \int_{\theta_0}^{\pi} \sqrt{\frac{2r^2(1-\cos\theta)}{2rg(\cos\theta_0 - \cos\theta)}}\, d\theta = \sqrt{\frac{r}{g}} \int_{\theta_0}^{\pi} \sqrt{\frac{1-\cos\theta}{(\cos\theta_0 - \cos\theta)}}\, d\theta \qquad \text{Eq 2.2.19}$$

Implementing the trigonometric double angle double-angle formula allows further manipulation:

$$t = \sqrt{\frac{r}{g}} \int_{\theta_0}^{\pi} \frac{\sin\frac{\theta}{2}\, d\theta}{\sqrt{cos^2\frac{\theta_0}{2} - cos^2\frac{\theta}{2}}}\, d\theta \qquad \text{Eq 2.2.20}$$

Approaching the above integral with the reverse chain rule or integration by substitution allows it to be computed:

$$u = \frac{\cos\frac{\theta}{2}}{\cos\frac{\theta_0}{2}} \qquad , \qquad du = \frac{-\sin\frac{\theta}{2}}{2\cos\frac{\theta_0}{2}}\, d\theta$$

Inputting the above expressions then gives:

$$t = -2\sqrt{\frac{r}{g}} \int_1^0 \frac{du}{\sqrt{1-u^2}} = 2\sqrt{\frac{r}{g}}\, {}_0^1[\sin^{-1} u] \qquad \text{Eq 2.2.21}$$

$$t = \pi\sqrt{\frac{r}{g}}$$

Both instances yield the same time of $t = \pi\sqrt{\frac{r}{g}}$, proving that the time taken to reach the bottom of the Brachistochrone is independent of where any sphere may be realised from on the curve [11].

2.3 Background into Blender and free body simulations:

Blender is an open-sourced and free downloadable 3D computer graphics software toolset. It contains a wide range of facilities to create visual effects, animated films, interactive 3D simulations, 3D printed models, computer games, and much more. For the majority of this project, the focus was around Blenders simulation functions and ability. The capacity that allows for Blenders 3D interactive applications comes in the form of the Bullet Physics Engine (*Bullet Physics*) [12].

The process involves setting up properties on the specific objects in the environment. For example, a body's dimensions, mass, and other components such as its coefficients of friction and restitution. Objects within the simulation can be seen to be either active or passive. Active meshes are subject to gravity and move freely corresponding to any forces that are applied, while passive bodies are essential bolted down and act only as a collision surface. The next step involves simply letting the engine take over as gravity is enabled and the objects interact with one another. These simulations produced by the physics engine are used commonly in computer games but can also be used for animation purposes [13].

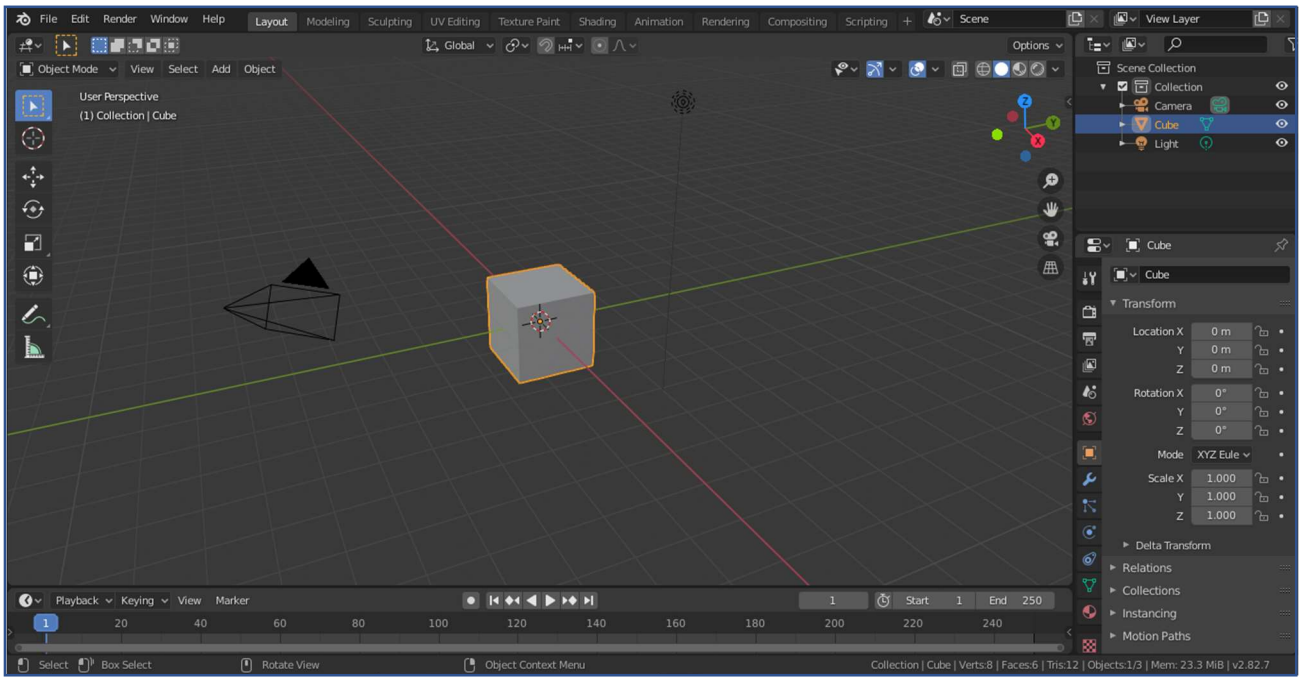


Figure 2.3.1- User interface for Blender 3D software

2.4 Context linking the project and Digital Learning

This project has critical ties linked to improving research and teaching as outlined in the project description. Higher education is facing several challenges that affect teaching and learning within universities. Student numbers are increasing, while the staff/student ratio continues to decrease. Therefore, there is a constant demand for increased flexibility when it comes to how learning opportunities can be provided and their effectiveness. To respond to the changes in student population it has become relevant to provide a more integrated use of digital technologies across universities and update their pedagogy. Pedagogy is the method and practice of teaching and everything it encompasses. Educators must develop their digital skills and adapt to more innovative teaching approaches to evolve as time goes on for all the reasons listed above. Digital learning in conjunction with project-based learning (PBL) [14] can serve a purpose in enabling more effective interaction between the content being thought with students. PBL involves teaching methods in which students gain skills and knowledge by working for an extended period to investigate and respond to an engaging and complex problem or question. Technologies such as Blender can provide a platform to greatly put more emphasis on the continued growth of all compasses of digital and project-based learning if certain educators decide to explore its capabilities. Similarly, it also encompasses more innovation for universal design for learning (UDL) [15]. UDL refers to a set of principles for curriculum development that gives all students an equal opportunity to learn. Applied effectively it provides a framework to improve the learning experience of all individuals within the mainstream teaching environment.

The *Irish University's Association* [16] has recently introduced a new three-year project aimed at enhancing the digital attributes and educational experiences of Irish university students through enabling technologies such as Blender. These practices demonstrate the capability and potential benefits of exploring these dynamical systems and how they could aid teaching and research across the entire UCD curriculum.

CHAPTER 3:

Overall Methodology.

3.1 Galton Board investigation:

When this project was assigned, the initial step was going through the project title in detail and grasping a good understanding of what the objective of the project entailed. Once the background surrounding the Galton board was understood while simultaneously undergoing tutorials online and becoming familiar with the software, the first developments of a simulated model were able to commence.

Initial simulations were run using square-based geometry for both the falling particles and the scatter pegs. This allowed the simulations to be generated and computed faster as there were fewer faces of contact for each collision a rigid body underwent.

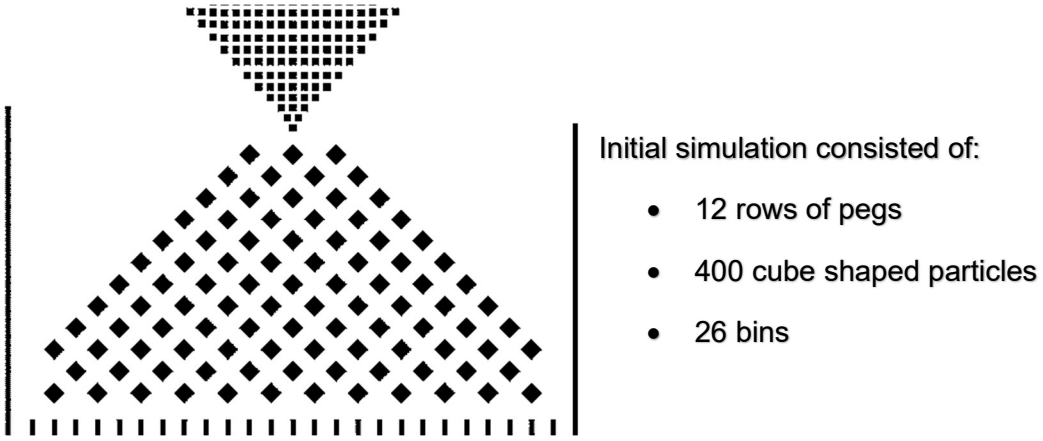


Figure 3.1.1- Initial virtual Galton board configuration

After several tests were conducted, both the friction and coefficient of restitution settings were refined throughout until an adequate gaussian form could be observed at the bottom of the simulated board. This was a promising start. The next step was to move to a more scripted based approach for conducting these simulations.

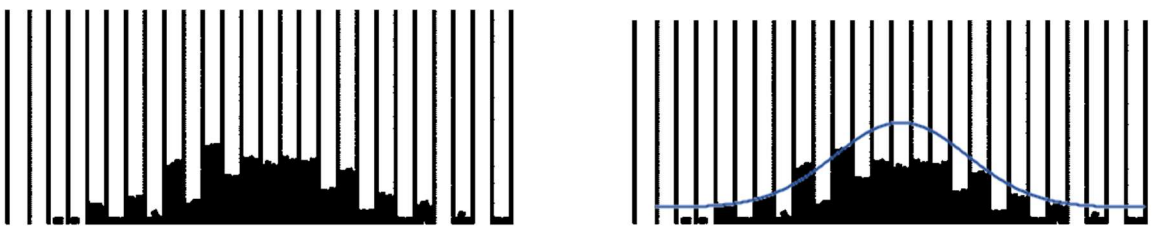


Figure 3.1.2- Final distribution of the initial square particle simulation, with an imprinted similar gaussian representation

Over several weeks, python code was written to generate any configuration of a Galton board after the code is run. The workings of the code employ Blender's scripting function to generate a mesh of specific dimensions. Each body's dimensions are dependent on the main six parameters inputted to the code. These initial inputs for the generation of a Galton board were as follows.

- Diameter of the scatter pegs
- Distance between pegs in x-direction
- Distance between pegs in z-direction
- Number of rows of pegs
- Number of slots
- Diameter of the falling particle

```
s = 0.45    # Diameter of the scatter pegs
gapx = 1    # Distance between pegs in x direction
gapz = 1    # Distance between pegs in z direction
n = 21      # Number of rows of pegs
slotn = 50 # Number of bins
cube = 0.4 # Diameter of particle
```

Blender script screengrab 1.1-Example code inputs

These parameters generate the dimensions and locations of each rigid body in the Galton board simulation by employing several equations and manipulations within the code. Two example input configurations can be seen below.

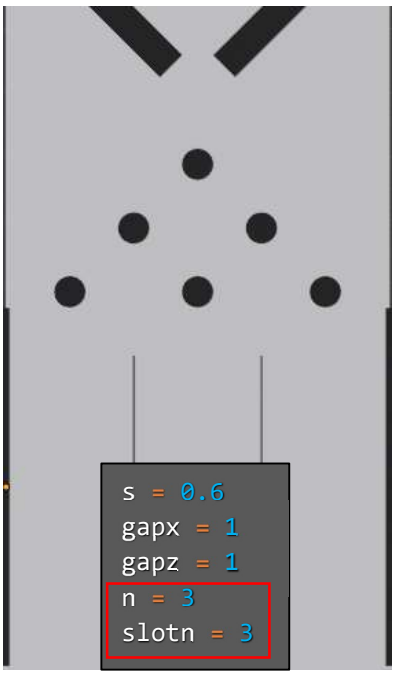


Figure 3.1.3- Configuration example A

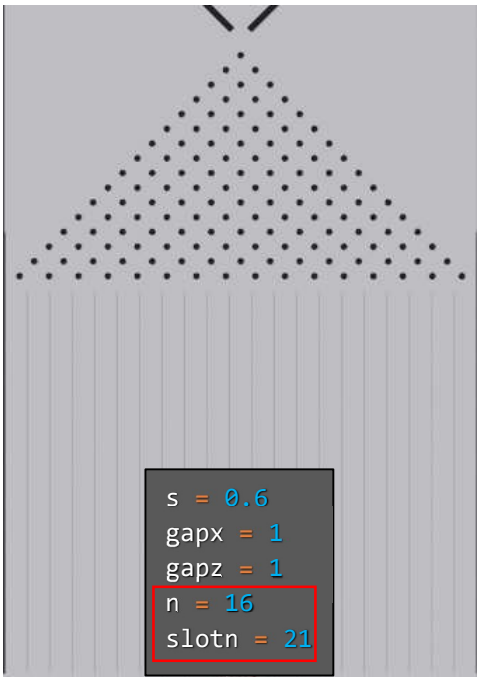


Figure 3.1.4- Configuration example B

Figure 1 shows the Galton board configuration where the inputted parameters for rows of pegs and slots are both 3 respectively. A quite miniature style design. Figure 2 shows the configuration when only these parameters are increased to a value of 16 rows and 21 slots. This shows the versatility within the script and how any combination can be constructed at the user's demand within seconds.

The bulk of this script computes by simply generating specific centre point and dimensions of each rigid body. Each body is passive and stays stationary throughout the simulation excluding the free-falling particles. Rigid bodies that only require a singular version of that geometry can be computed on by the inputs alone. The example shown below is the generation of the main base of the board.

```
l = n*gapz*3.75 #length of the board
w = gapx*n*2    #width of the board
bret = cube*1.05 #breadth of the board
ramp = l*0.5    #length of the ramp

x = 0           #Default x position
y = -bret*0.5   #Default y position
z = l*0.7       #Default z position
```

Blender script screengrab 1.2- Several calculated constants used throughout the

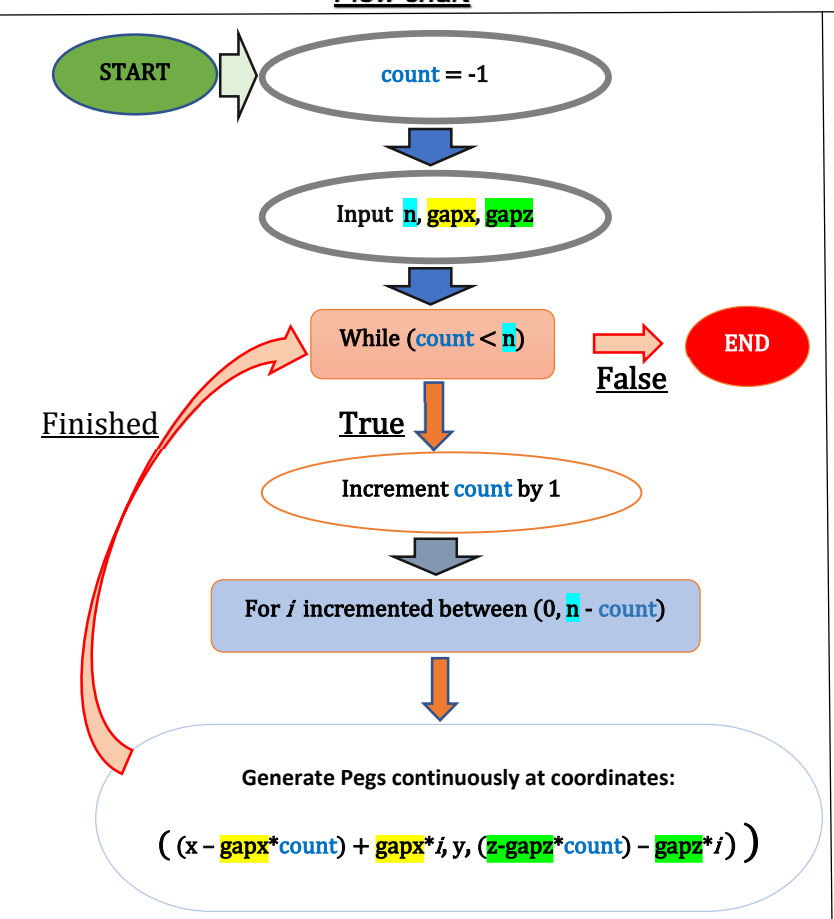
```
#Code that constructs the back board
bpy.ops.mesh.primitive_cube_add(size=1, location=(0, bret*0.5, l*0.5))
bpy.ops.transform.resize(value=(w, bret, l), constraint_axis=(True, True, True))
bpy.ops.rigidbody.object_add()
bpy.context.object.rigid_body.type = 'PASSIVE'
```

Blender script screengrab 1.3- Selected example of code to construct a rigid body

However, certain rigid bodies are required to have multiple different centre point locations of the same geometry. To achieve this, loop functions are required to determine each location of these multiple rigid bodies. One example is the code to generate the scatter pegs. It is also more complex as it requires a loop within a loop to construct them most efficiently. The original script for generating the pegs quincunx pattern can be shown below. A flow chart on the following page breaks down the specific operations step by step.

```
count = -1

while (count < n):      # Loops that generate the number of scatter pegs configuration

    count+=1

 for i in range (0, n-count):
  bpy.ops.mesh.primitive_cylinder_add(radius=s*0.5, depth=bret, location=((x - gapx*count) + gapx*i, y, (z - gapz*count) - gapz*i))
```

Blender script screengrab 1.4- Code to generate the formation of scatter pegs



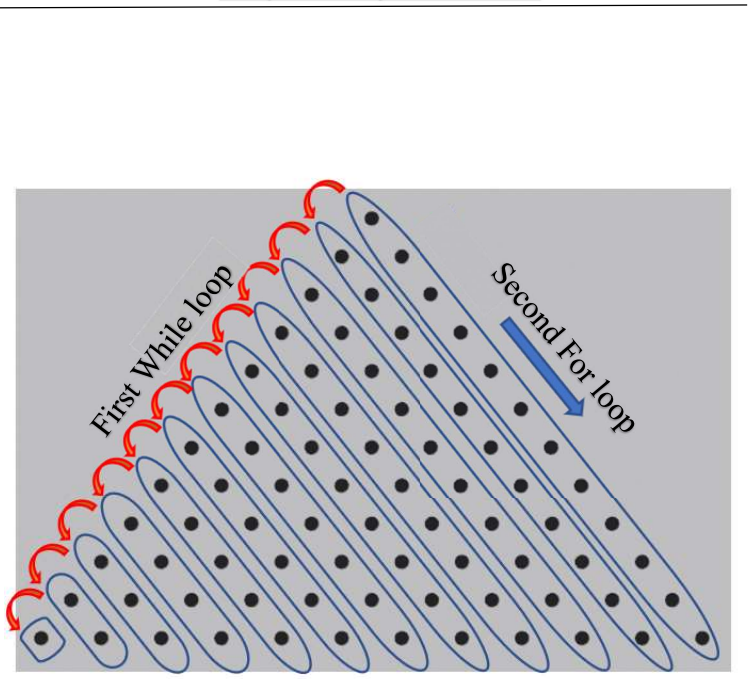


Figure 3.1.5– Example flow chart representation (n = 12)

The generation of the falling particles also employs a two-loop system. Although the specific starting location of each particle is not important, the loops are beneficial at generating the large number of rigid bodies that are needed, upwards of a thousand on most simulations.

Once the first version of the script was written, running several simulations was the next step. At the beginning of these initial simulations, it was decided to keep a quincunx pattern for the pegs. Meaning the input of the x and z distances between the pegs would remain equal. For the purpose of not taking extraneous amounts of time simulating and rendering, the number of rows of pegs was also kept constant at the adequate value of 12. This number is applicable as it gives a sufficient number of paths each particle can take. The value is also congruent with the majority of experiments Francis Galton conducted back in the statistical model's infancy and also the number of rows the physical model had that was purchased for this project [6].

The first rendered simulation consisted of 800 particles and 28 slots. The final frame of the simulation can be seen in the figure below. The settled particles Gaussian form is evident which was promising to see. Although not perfect, some interference occurred with certain pegs being situated directly above corresponding slots. This caused peak locking to occur in the Gaussian curve. However, this was addressed as the simulations progressed further.

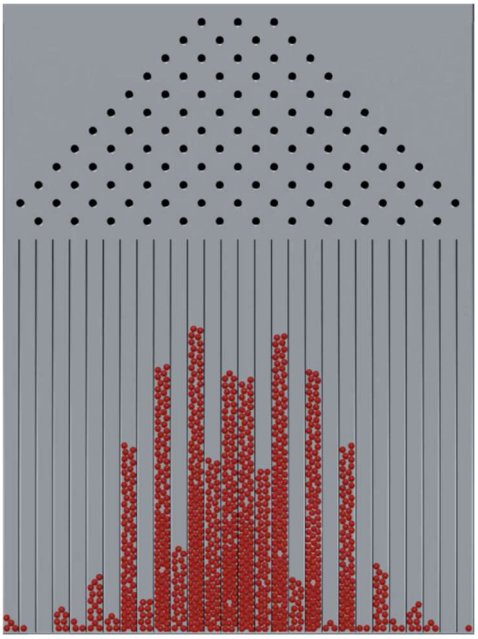


Figure 3.1.6- Gaussian distribution of initial scripted simulation

Going back to one of the roots of the project brief, it was outlined to investigate a ***biased*** Galton board. The next simulations were strides in this direction. It was found by adjusting values for the input peg diameter within the code yielded a change in the *standard deviation* σ of the settled gaussian curve. Example simulations employing these adjustments can be shown in the figures below. Both using an equal amount of 1000 particles.

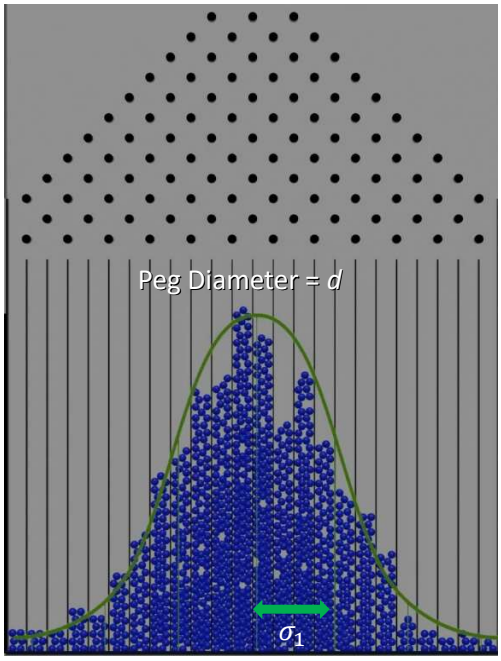


Figure 3.1.7- Distribution with narrower standard deviation

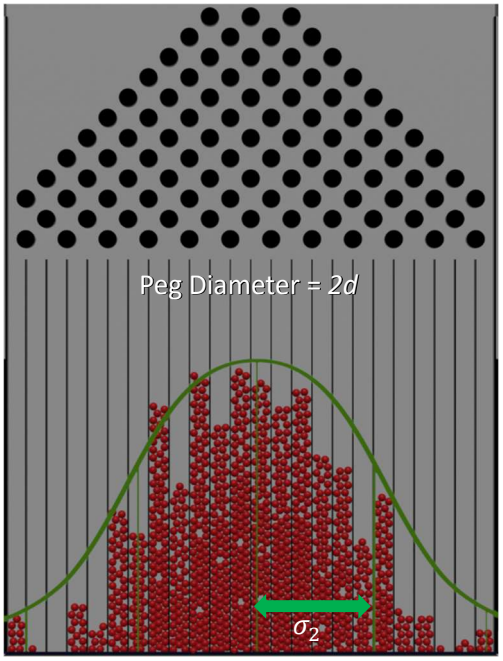


Figure 3.1.8-Distribution with wider standard deviation

Further efforts to achieve more bias within the user's control was the next priority. The approach was to divide the variable that controlled peg separation in the x-direction into two separate components. One being the gap distance to the left, and gap distance to the right. Introducing these parameters allows the user to add bias and shift the eventual settled gaussian to one side or another. Dependent on the ratio of the subsequent pegs triangular shape and their angles of pitch. While in the quincunx pattern the pegs in the previous simulations formed an equilateral triangle. Now the user can adapt the pattern to form a multitude of different scalene triangles that can produce various final form outcomes.

Achieving this was done by simply breaking down the previous variable into two. Once each variable denoted to separate left and right components, the equation seen previous that generates the location of each peg only needs to be altered slightly to accommodate this.

```
lgapx = 1   # Distance between pegs to the left

rgapx = 1.25  # Distance between pegs to the right

gapz = 1  # Distance between pegs in z direction
```

Blender script screengrab 1.5- Previous input broken down into two components example

```
location=((x - lgapx*count) + rgapx*i, y, (z - gapz*count) - gapz*i))
```

Blender script screengrab 1.6- component that moves loops in either direction implemented

In the previous script, both loops had the same gap distance in the x-direction. Making these changes as seen in the figure above allows the shift in the pattern of pegs to take place.

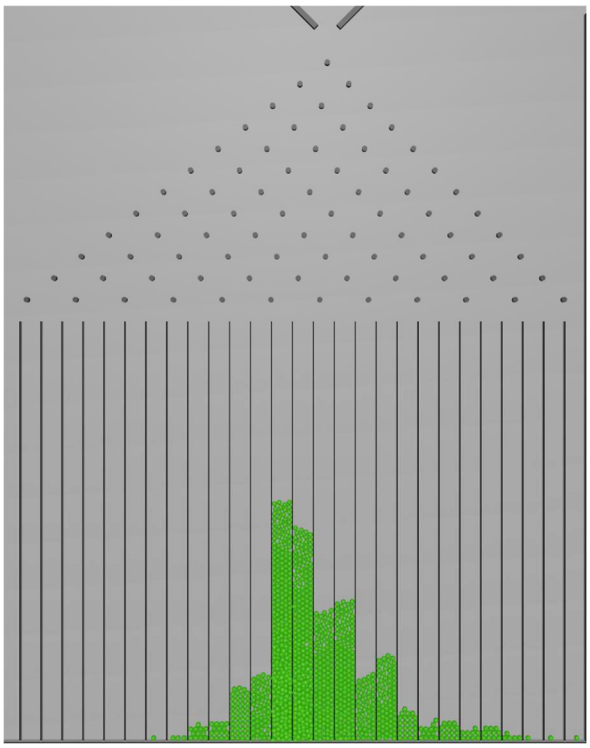


Figure 3.1.9- Skewed distribution to the right

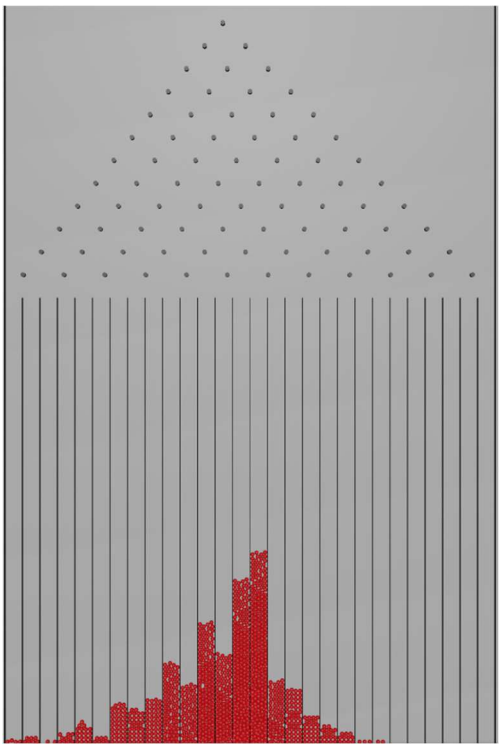


Figure 3.1.11- Skewed distribution to the left

Simulated tests with these alterations showed promising results as seen in the two figures above. The biased pattern of pegs caused the particles to form a remarkably similar form to that of a *lognormal distribution*, instead of the Gaussian distribution seen in previous simulations. At these proceedings, the final particle distribution can only be observed, not analysed. Therefore, it was not applicable to perform certain statistical tests. With these advancements in creating controlled bias can be shown to be significant steps of progress to regards to what was laid out in the project description. Further efforts to generate a more adaptive programme that can analyse and represent the particle distribution in terms of a function were the next steps.

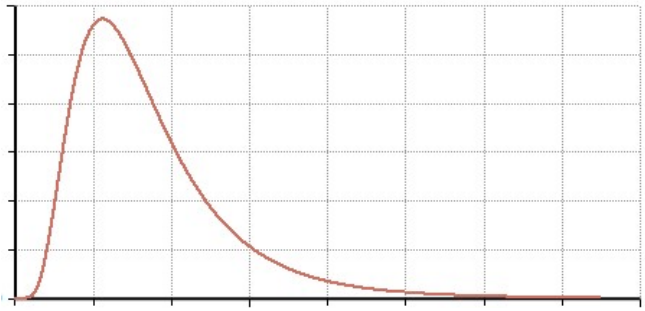


Figure 3.1.12- Lognormal distribution graph [17]

Blender's capabilities to investigate the various particle distributions is limited. Therefore, it is required to use *MATLAB* in conjunction with Blender to conduct a suitable analysis. For this, an additional script was developed that successfully exports the coordinates of each particle to a CSV file.

```
bpy.ops.object.select_same_collection(collection="spheres")
bpy.ops.object.visual_transform_apply() # Applies visual transformation

os.chdir('C:\\Users\\user\\Desktop')                    # File directory
with open('xznorm.csv', 'w') as csvfile:              # File name
    writer = csv.writer(csvfile, delimiter=',')
    for sphere in bpy.data.collections['spheres'].objects: #Loops through collection of Spheres
        writer.writerow([sphere.location[0],sphere.location[2]]) # Records x and z position
        sphere.location[2]
```

Blender script screengrab 1.7- Python code that creates csv file of particle data

The script is intended to run after the particles have settled at the end of a simulation. It first applies a visual transformation to the created collection of sphere particles to set the correct coordinates at that the last frame. The Y coordinates are not recorded as is unnecessary with all values being equal.

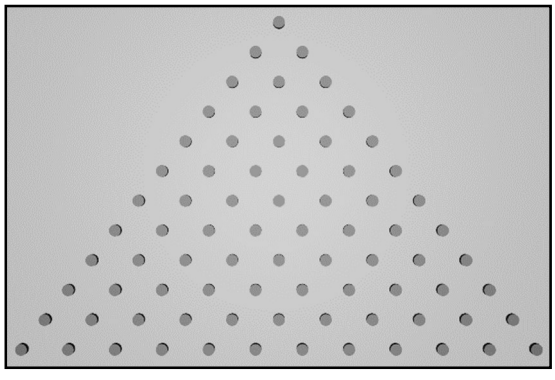
An identical approach is taken in an additional script that creates a CSV file of the location of the collection of bins. The x coordinates are the only row that is required. Subsequently importing these obtained data sets into MATLAB is the following step.

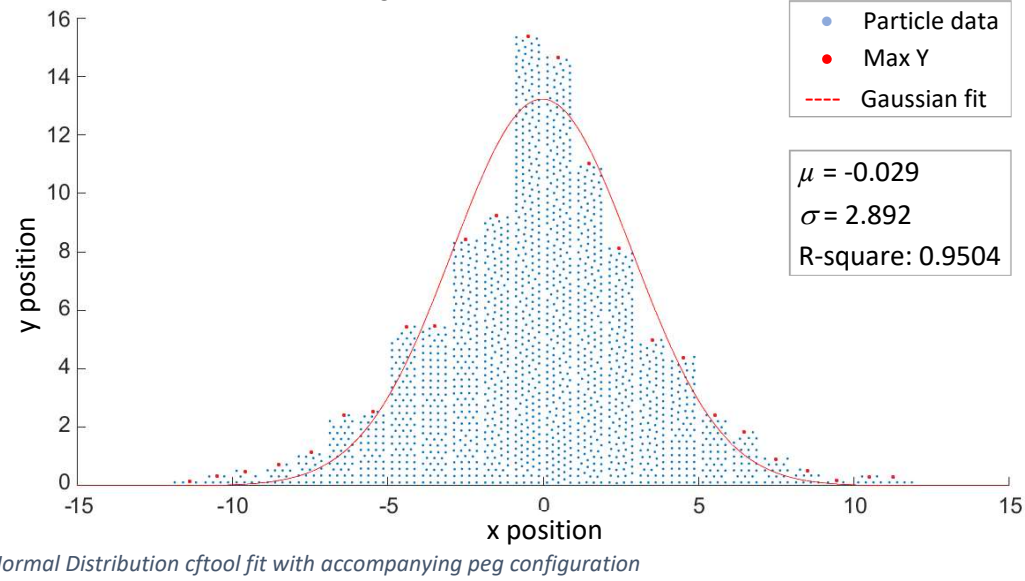
The drafted MATLAB code shown reads in the CSV files by name and makes it available in the workspace area. Implementing a for loop it goes through each specified bin and finds the particle with the max y value within that range. The corresponding x value for these max y coordinates is simply the halfway point between each bin.

```
Array=csvread('xyznorm.csv');  % Reads in particle data
x = Array(:, 1);
y = Array(:, 2);

plot(x, y,'.')                 % plots particle coordinates

binx = csvread('xyznormbins.csv');
binx = sort(binx);             % Reads in and sorts bin locations

for n = 1:length(binx)-1       % Loops through each bin
    try
        ymax(n)  = max(y(x >= binx(n) & x < binx(n+1))); % Max y value in that range
        xmean(n) = mean(x(x >= binx(n) & x < binx(n+1))); % Midpoint of range
    catch
        ymax(n)  = 0;
        xmean(n) = 0;          % Intercepts null values of 0
    end
end
```

MATLAB script screengrab 1.1- Drafted code which plots max bin value

Having obtained these max coordinates from the data each point can be analysed within the curve fitting tool MATLAB provides. This allows for a statistical representation of the distribution to be applied. Hence, for the sample with normally distributed data shown below properties such as mean, standard deviation, and overall goodness of fit can be retrieved.



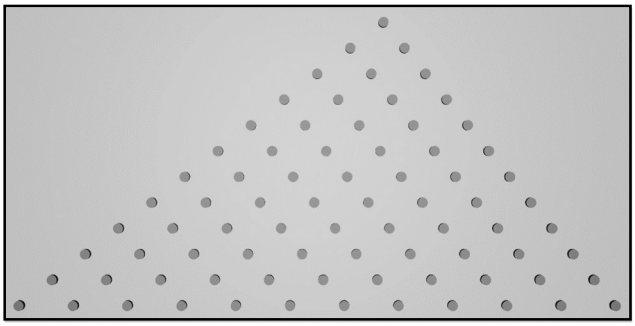


MATLAB fig 1.1- Normal Distribution cftool fit with accompanying peg configuration

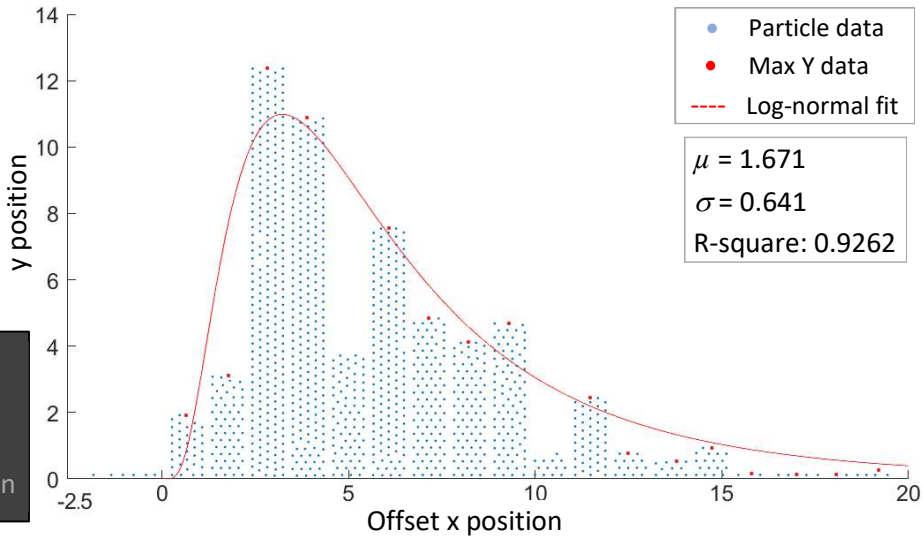
This information indicates the extent of what the simulation inputs had on the final particle distribution. The above sample it tells us that it is has a quality goodness of fit to a gaussian distribution. Which is expected for a peg configuration in a quincunx pattern.

Similarly, analysing distributions of a shifted peg variation the final particle coordinates the same process applies. From there it is applicable to test the hypothesis if the data does indeed resemble that of a log-normal distribution. The MATLAB script once again pinpoints the max bin locations, but it had to be noticed that simulations conducted with a bias tended to experience peak locking in certain bins. For this reason, to not shift the fitted curve of the distributions outline it was deemed appropriate to not acknowledge these points. Another small subset of outliers at the left tail of the data was also excluded to unbalance the overlaid curve.

To then successfully apply a log-normal fit it was also required to add a line of code which offsets each x coordinate to be all positive. The shape of the distribution remains the same but is an intended step as it is not theoretically possible for a log-normal distribution to allow negative values. When Implementing these additions to the code a sample analysed biased distribution can be seen below. The ratio of which the gap in the scatter pegs to the left compared to the right was set at 1.6:1.



```
lgapx = 1.6 #Gap of pegs to the left
rgapx = 1   #Gap pegs to the right
gapz = 1.27 #Gap between pegs in z direction
```



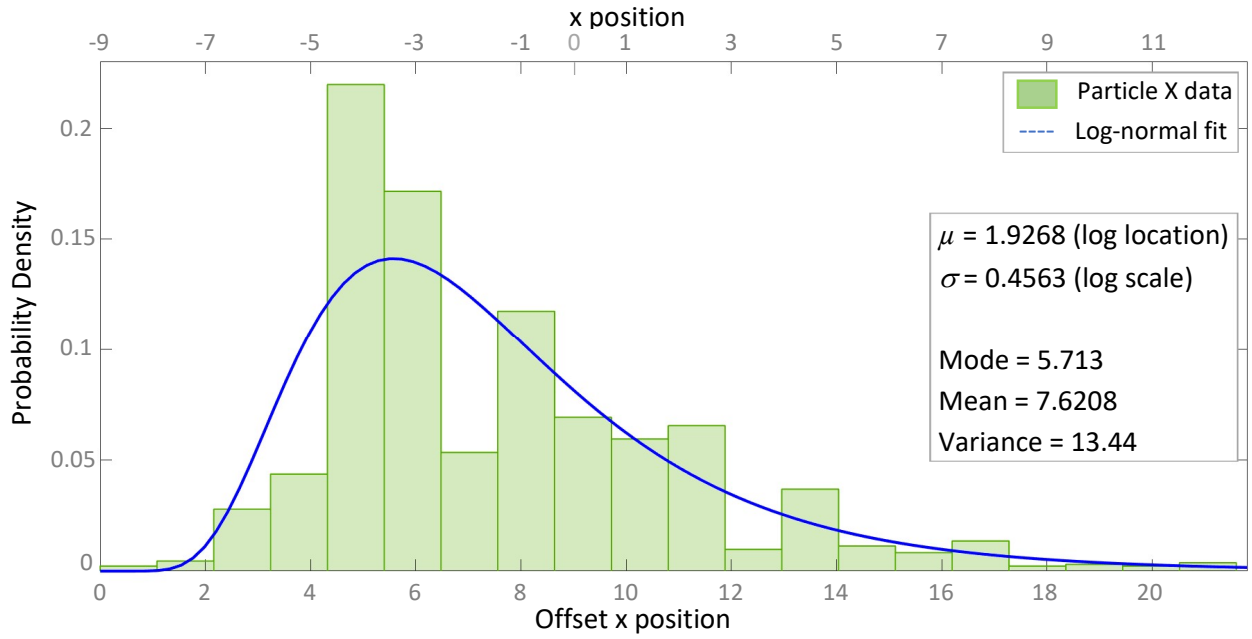
MATLAB Fig 1.2- Log-normal Distribution cftool fit with accompanying peg configuration

The model generated by the curve fitting tool showed promising results, one value of note was an R-squared of 0.9262. This value is by no means the deciding factor with regards to adopting the distribution as log-normal. However, it tells some capacity that the outline distribution is, in fact, congruent with the premise.

To further test the robustness of any connection it would be beneficial to conduct more clinical statistical tests. An additional approach would be to convey the data as a probability density function (PDF), opposed to being described by its y coordinates. A PDF is a statistical expression that defines a probability distribution, where the area underneath the curve represents the likelihood within a given range. Adjusting elements in the previous code with also the addition of MATLAB distribution fitter a pdf can be achieved.

```
% Log normal is always positive so offset data
min_val = ceil(abs(min(X)));

% This offsets the data to be all positive.
bins = bins + min_val;
X = X + min_val;

pd = fitdist(X,'lognormal'); % This fits a log-normal pdf

histogram(X,bins,'Normalization','probability')
hold all  % Plot a normalised histogram with the bins as in xyzbins.csv

% Array to cover the range of the data for plotting the fitted PDF.
x_values = linspace(min(X),max(X),64);

y = pdf(pd,x_values);           % This gets the PDF curve from pd and x_values
plot(x_values,y,'LineWidth',2) %  plots data
```

MATLAB script screengrab 1.2- Drafted code which plots max bin value



MATLAB Fig 1.3- Log-normal fitted probability density function

Useful attributes within the FITDIST app to allow for parameters such as mean and standard deviation to be available and displayed. This above fitted distribution does consider all x variables including the data of the peak locked bin. Therefore, while providing a worthy representation of the distribution set it is in turn slightly shifted downwards in comparison to the previous fitted log-normal curve using the max bin location data.

Now having obtained the particle data as a probability density function more parameters and means for analysis are available. The non-offset X positions were added to the plot to distinguish between factors in the actual ran simulation. Having returned a mode of 5.713, this corresponds to a value of -3.287m. The particles were released from an x position of 0m, marked on the figure above. This tells the extent to which μ can be altered when a bias is introduced. This can be interpreted to mean the probability of a particle taking a left and right bounce are not equal. This can be demonstrated by observing the trace of three randomly selected particles shown in the figure below.

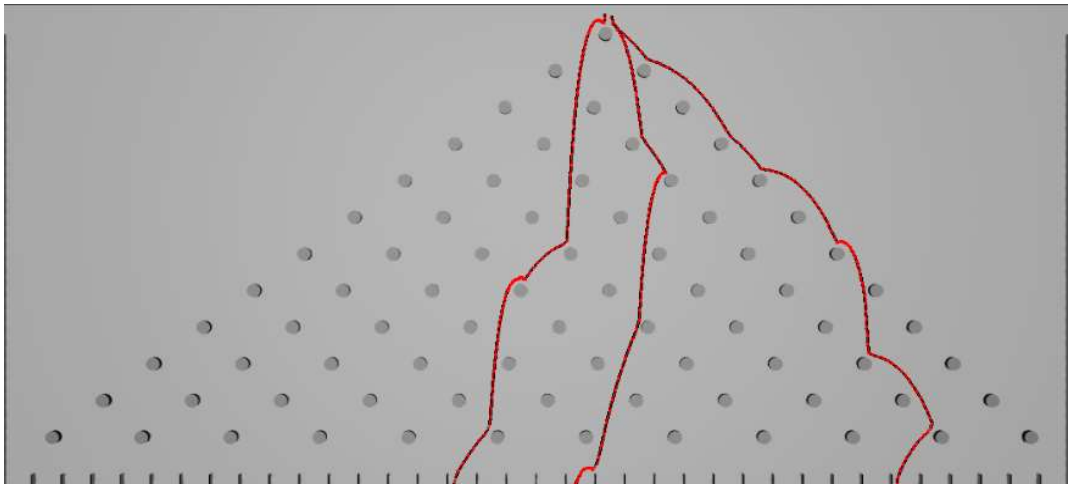
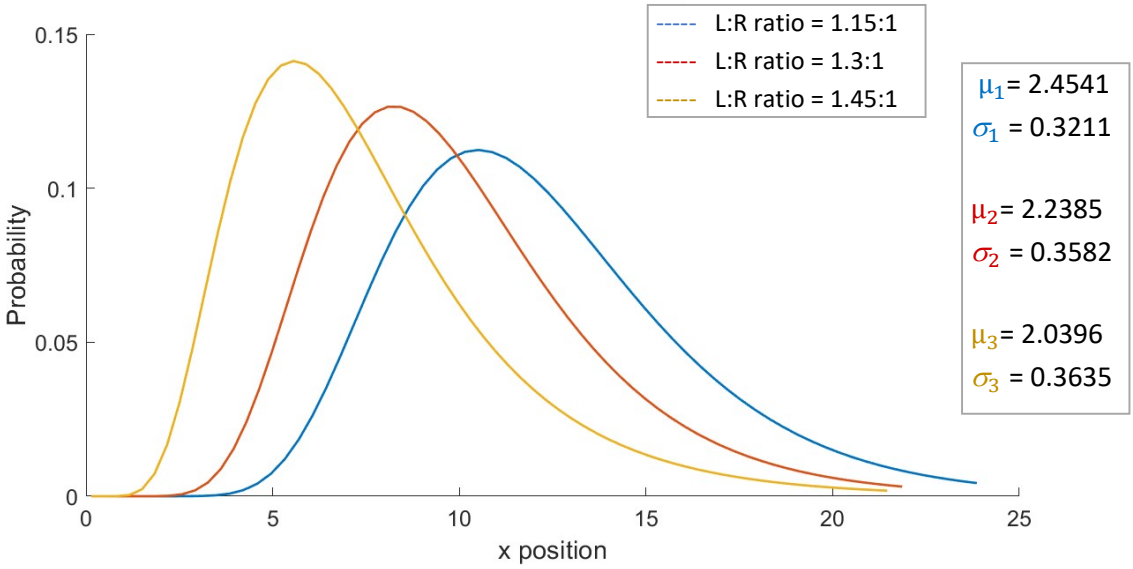


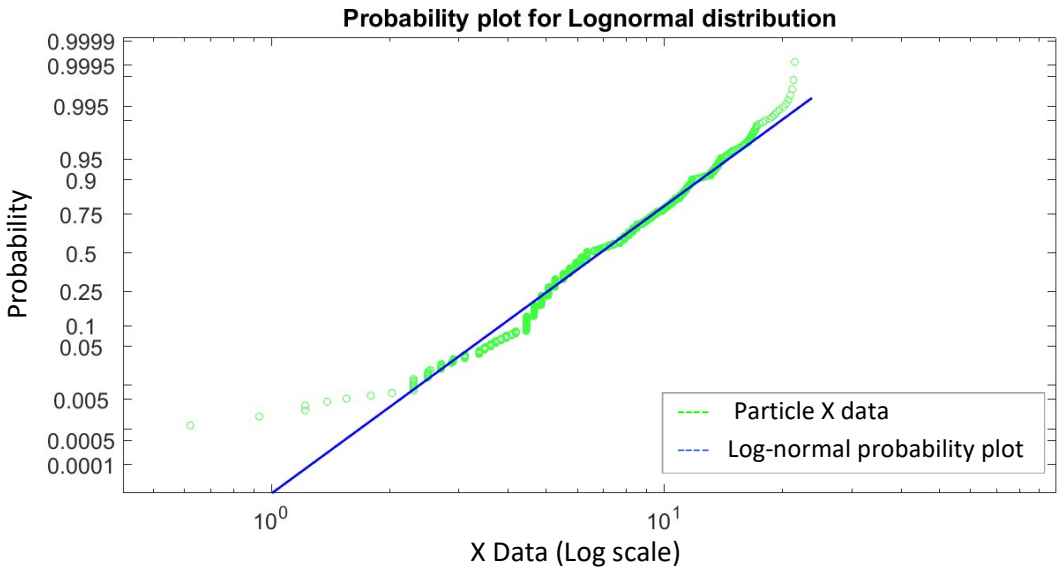
Figure 3.1.13- Traced path of randomly selected particles

It shows once a particle takes a left bounce, the odds of repeating a bounce to the left are perceivably more likely. However, as distances between pegs to the left substantially greater it is not expected for particles to reach the extreme end of bins situated on the left. Contrasting probability density functions of conducted simulations where this distance between scatter pegs is constantly increased solidifies these outlooks on a controllable shift on the final distribution.



MATLAB Plot 1.4- Comparison of probability density functions of increasing bias parameter

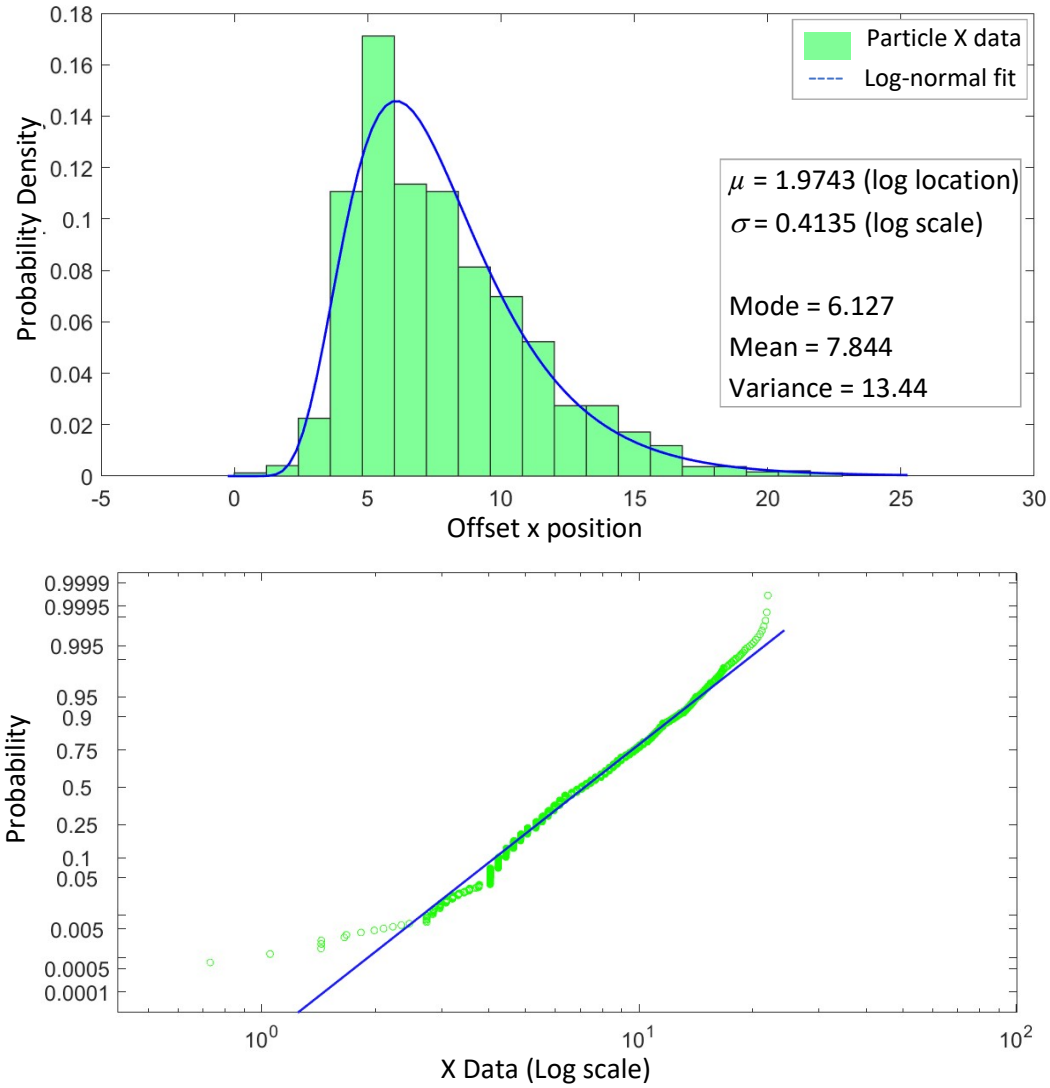
It can be said that the simulation containing the inputs of 1.15:1 more resembles a gaussian form. However, it is clear there is a transformation to a more log-normal outline evident as the ratio parameter is increased. Additional analysis available to test the validity of these findings would be a probability plot of the data. This will shed light on what areas of the distribution fit well under log-normal conditions. Reverting to the previous PDF of the conducted simulation entailing a 1.6:1 peg ratio, the probability plot gives the following.



MATLAB Plot 1.5- Log normal probability plot vs distribution data

The graph tells that the tails of the distribution are incompatible with the probability function. However, a large majority of the data lies within the boundaries of the fitted plot. The peak of the data can also be seen to not fall in-line with the expected regression. This could be attributed to over a large number of particles involved within the simulation this bin location

would be most likely to experience an influx of overflowing particles. One method to improve the alignment for the plot could be to further refine the collection of pegs within the simulation. Up until this moment, all simulations have been conducted using 12 rows of scatter pegs. If this were to be increased, it would introduce a significant number of further paths each particle could take to reach a specific bin. Therefore, increasing the rows of pegs to a value of 16 for a follow-up simulation while keeping the number of particles the same produced the following distribution.



MATLAB Plot 1.6- PDF & Log normal probability plot for increased number of rows (n =16)

This addition of further rows leads to a more congruent fit in both the PDF and the probability plot. Again, the only offset data situated at the tails of the distribution and its peak. However, the alignment can be seen to have improved. Overall, both fits can be seen as a fair indication

that the distribution is not exclusively linked with that of a log-normal distribution but entails a noteworthy resemblance in certain ranges.

The final approach looking into the Galton board involves exploring is to implementation of a peg configuration void of bias. In simpler terms a complete random generated sequence. These methods will be interesting to adapt to the existing Galton board script to test the effects of a somewhat random scatter peg variation on the final particle distribution. Most of the previous script can be availed of, adjustments only need to be made to the scatter peg generating loop.

To implement these changes many new parameters must be introduced. The most essential of these being the specific number of scatter pegs, and the approximate area of which pegs are situated. It was also important to include an additional factor that prevents pegs forming less than that of the diameter of a particle. Providing this element ensures no particle will get lodged within the scatter frame. The drafted script below then enables the random generation of pegs to the user's specific preferences.

```
for i in range(num_circles): # Loop that runs through the specified number of scatter pegs
    j = 0
    searchOn = True
    while searchOn:
        if j > max_tries:    # Max iterations finding peg location
            print("Placed", i, "of", num_circles)
            bpy.ops.object.select_all(action='DESELECT')
            ref_circle.select_set(True)
            bpy.ops.object.delete()
            raise ValueError('No space for another circle')  # Gives error if location is void
            break
        j += 1
        new_location = (xminm + random() * sx, y,
                        yminm + random() * sy
                        )                                    # Generates random reference location

        for existing_location in existing_locations:
            if (Vector(existing_location) - Vector(new_location)).length < (2 * radius + cube*fact):
                break               # Breaks if location is less than the specified gap between pegs
        else:
            new_circle = ref_circle.copy()    # Denotes location once it is found
            new_circle.location = new_location
            sce.collection.objects.link(new_circle)
            existing_locations.append(new_location)
            searchOn = False                # Restarts system to start the process over again
```

Blender script screengrab 1.8- Python code that creates random scatter peg configuration

For an arbitrary scatter pegs number and a gap factor value of 1.5 gives a random configuration the likes that can be seen below.

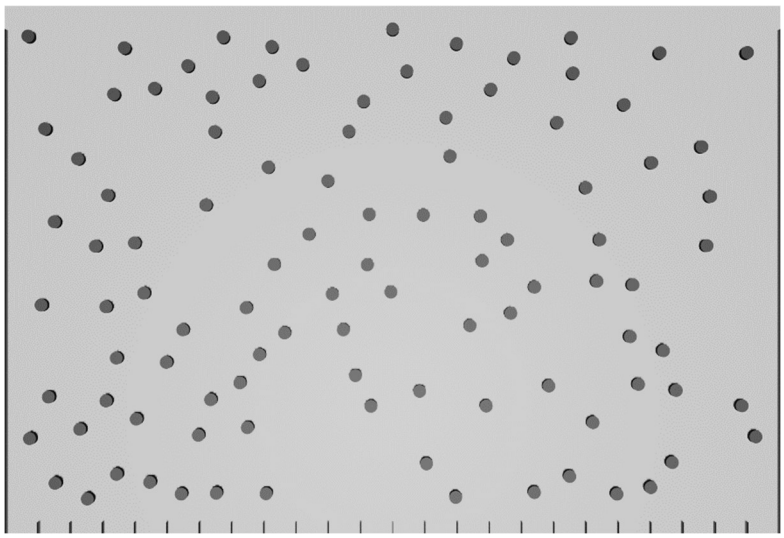
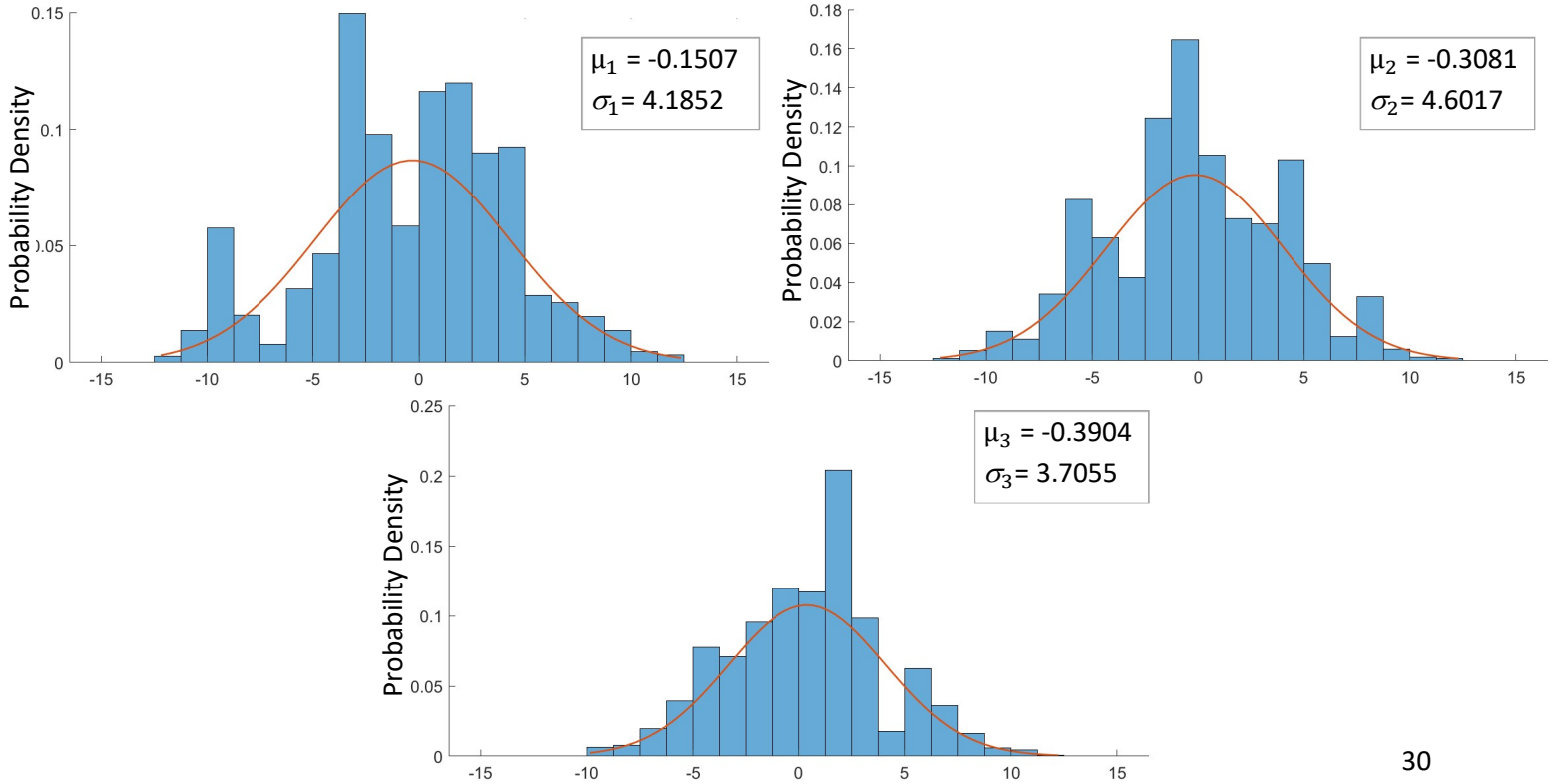


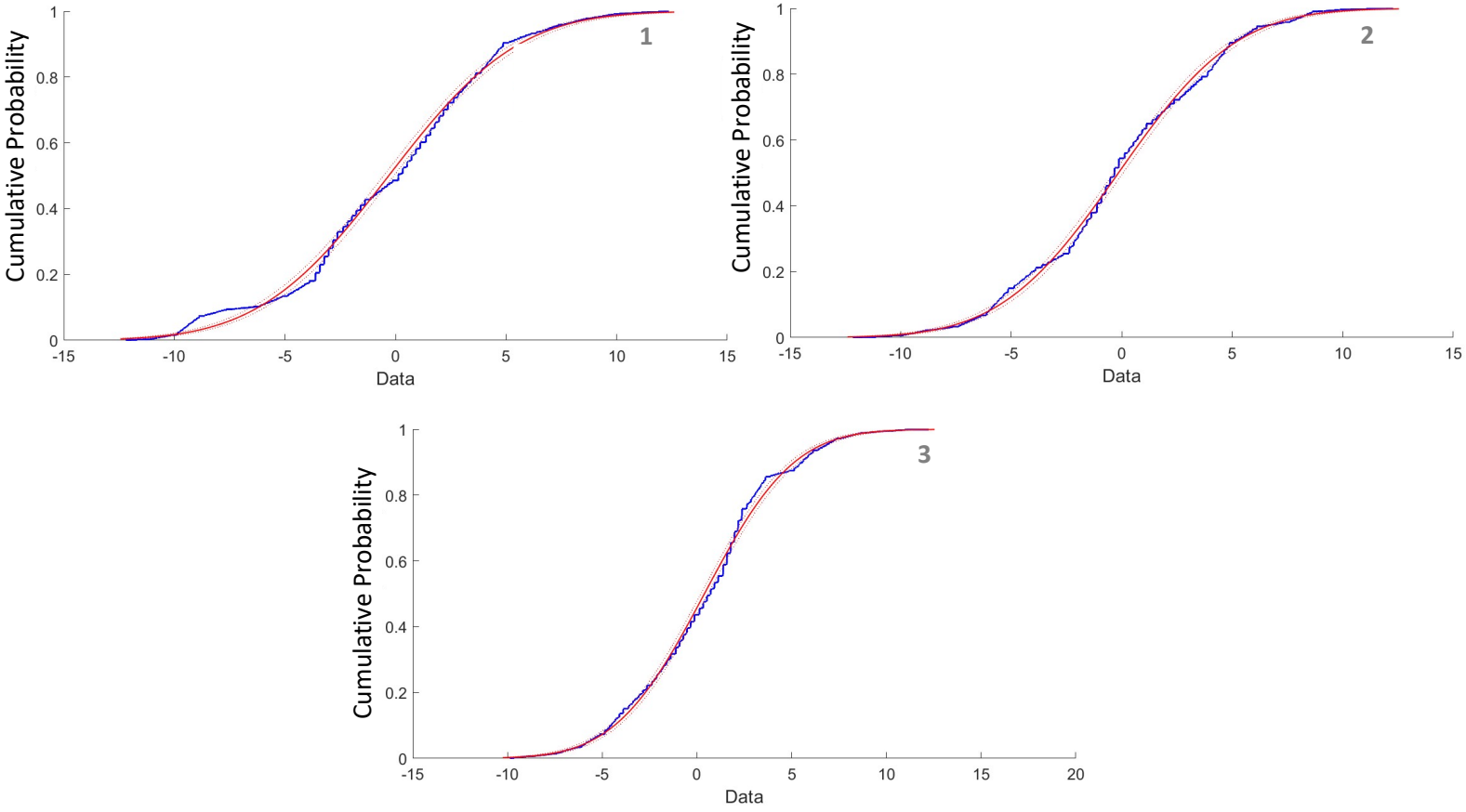
Figure 3.1.14- Created array of random scatter pegs

The only non-random location is a default peg that is situated directly below the opening the particles are released. This at least ensures an even distribution of particles for both the left and right. With regards to conducting test simulations, it was decided that at first a value of 100 pegs would be adequate. For the number of particles used in the simulation 1500 was chosen. This will remain constant across three conducted simulations. Once the tests were performed, the same developed MATLAB code used to visualize the distributions as a probability density function can be deployed.



MATLAB Plot 1.7- Distribution comparison for simulations of a random peg variation

The results show when the data is represented as a normal distribution, each does display promising similarities. All three simulations produced a mean which is considerably in the proximity of one another, while also not distant to absolute 0 which would be expected for an ideal normal distribution. It is apparent that some bins experienced peak locking leading to various dips and peaks within the data. Similarly, these phenomena can still be seen for various distributions resulting from the regular quincunx peg pattern. Therefore, while the peg variation can be classified as random, the settled particle shape each produced would seem incorrect to be considered random. This can be further iterated by inspecting each cumulative distribution function.



MATLAB Plot 1.8- CDF comparison for simulations of a random peg variation

With regard to building upon these encouraging outcomes, many different routes could be taken. Following this chapter possible future research and relatable work will be discussed to see what other developments could be made.

3.2 The Brachistochrone problem investigation:

After coming to the end of proceedings exploring the Galton board, it was time to investigate the brachistochrone curve. The goal was to simulate and analyse both the brachistochrone and tautochrone aspects. Conducting these simulations of these particularly interesting concepts will hopefully shed light on these many different phenomenal attributes. Other efforts would be more difficult to achieve on a physical model. Having come a custom to Blender capabilities and functions progress was set to come more efficiently. To plot the brachistochrone within blender it was prudent to first revisit the equation for a cycloid that was explored in the technical section previously. For a curve beginning at an arbitrary point (x_1, y_1) and finishing at (x_2, y_2), the corresponding function is the following:

$$x = r(\theta - \sin\theta) + x_1 \ , \ y = r(1 - \cos\theta) + y_1 \qquad \text{Eq.3.2.1}$$

When these start and end points are known, the values for *r* and θ can be solved computationally. *r* is given as the radius of the traced circle and θ is the completed angle of rotation. Latest Blender versions does contain a *numpy* library but do not have the built-in function solving capabilities needed to solve for these values. However, with MATLAB's *fsolve* function the procedure can be completed. Therefore, the approach is to calculate the needed curve coordinates within MATLAB and export it to a CSV file that could easily be read in by Blender. The drafted code to acquire the coordinate CSV file can be shown below:

```
x1 = 0; % example starting x coordinate
y1 = 7; % example starting y coordinate
x2 = 11; % example final x coordinate
y2 = 0;  % example final y coordinate
xx2= x2-x1; % x2 position if x1 was 0
yy2= y2-y1; % y2 position if y1 was 0

% below function of the needed cycloid if (x1,y1) = (0,0)
f = @(b) [(b(1)*(b(2) - sin(b(2))) - xx2); (-b(1)*(1 - cos(b(2))) - yy2)];
b = fsolve(f,[1 2]); % solves for r and theta

r = b(1);                   %denotes radius
deg = round((b(2)*180)/pi);%converts theta value from radians to degrees

for n = 0: deg         %loop running from 0 degrees to theta degreees value
    rad = (n*pi)/180; % converts theta back to radians

    x(n+1) = r*(rad - sin(rad)) + x1; %calculates each x coordinate
    z(n+1) = -r*(1 - cos(rad)) + y1;  %calculates each y coordinate
end
```

```
x = x'; %changes x coordinates from row to a column array
z = z'; %changes y coordinates from row to a column array

zc = zeros(size(x,1),1); % gives a 0-array length x for the set y coordinates in
blender

xz = [x,zc,z]; % combines final coordinate list array
writematrix(xz,'curve117.csv') %writes csv file of curve coordinates
```

MATLAB script screengrab 2.1- Developed code which calculates curve coordinates csv file

Now having the capacity to acquire the cycloid coordinates in an easy to manage CSV file, the next step was to complete the blender script that would take this data and compile the system needed to run realistic real body simulations.

The workings of the drafted code are similar to that of the Galton Boards and employ blenders scripting function to generate a specific mesh. Each specific dimension of a body is dependent on the inputted coordinate file. There are also four additional parameters that the user inputs to the code on the basis preferences for the final simulation. These initial inputs for the generation of the simulation set up include the radius of the tracks and the sliding sphere. This initial section of the script assesses the CSV file and designates a vertex edge for each coordinate. Linking this data to populate the subsequent curve path. Depth is added to the curve with a BezierCircle with a radius corresponding to the user's input.

```
filename = 'curve117.csv'
directory = 'C:\\Users\\user\\Desktop'
fullpath = os.path.join(directory, filename)

verts = []                          # Read verts from CSV file and store in a float list
with open(fullpath, newline='') as csvfile:
    reader = csv.reader(csvfile, quoting=csv.QUOTE_NONNUMERIC)
    verts = list(reader)

edges = []                          # Compute edges from vertex list
for edge in range(len(verts)-1):
    edges.append([edge,edge+1])

Mesh = bpy.data.meshes.new("brachistochrone_mesh")  # Create empty mesh and attach to object
Obj = bpy.data.objects.new("brachistochrone_obj", Mesh)
Obj.location = bpy.context.scene.cursor.location

Mesh.from_pydata(verts,edges,[])                  # Use vertex and edge data to create the mesh
bpy.context.collection.objects.link(Obj)

ob = bpy.context.scene.objects["brachistochrone_obj"]   # Get the object
bpy.ops.object.select_all(action='DESELECT')            # Deselect all objects
bpy.context.view_layer.objects.active = ob              # Set brachistochrone_obj as active object
ob.select_set(True)                                     # Select the brachistochrone_obj
bpy.ops.object.convert(target='CURVE')                  # Convert to path to curve

# Create outline circle for curve bevel geometry
bpy.ops.curve.primitive_bezier_circle_add(radius=radius, location=(0, 0, 0))
bpy.context.object.data.bevel_object = bpy.data.objects["BezierCircle"]
```

Blender script screengrab 2.1- Python code that generates mesh from coordinate file

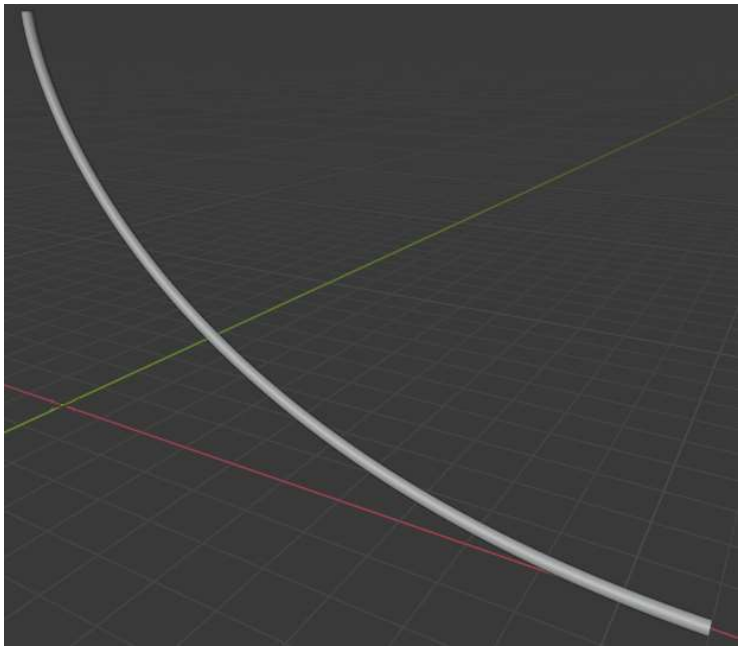


Figure 3.2.1- Developed brachistochrone mesh

With the desired mesh being achieved the intension of the remainder of the code is to test this optimal time curve against other paths. One of these being a linear path of shortest distance between points (x_1, y_1) and (x_2, y_2). The second being a path of steep curvature that allows the sphere to gain a substantial amount of initial velocity. However, in turn has a further distance to travel. All paths will employ a cylindrical geometry similar to the mesh seen above but will require a duplicate mesh of the path to act as a track for a sphere to slide down.

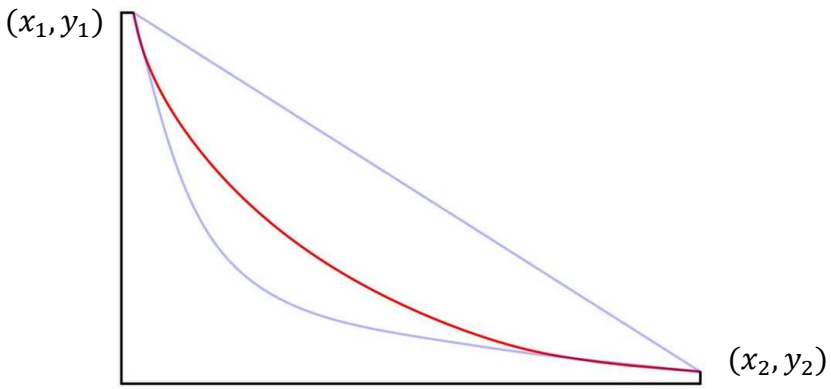


Figure 3.2.2- Test of the Brachistochrone and two other alternative paths [18]

Using the Pythagoras theorem, the shortest distance between the start and end positions can be calculated. The specific angle at which the track is to be pitched at is simply $\tan^{-1}(\frac{x2}{y1})$.

```
# concept code generates liner track from (x1,y1) -> (x2,y2)
bpy.ops.mesh.primitive_cylinder_add(radius= radius,depth = sqrt(x2*x2 + y1*y1),
                                    location=(x2/2 +x1, 2*gap, y1/2 ))
# Rotates mesh correct angle which tan inverse of (x2/(y1)
bpy.ops.transform.rotate(value=-math.atan(x2/y1), orient_axis='Y',
                                    orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0, 1)))
```

Blender script screengrab 2.2- Portion of script that generates mesh for linear track

The next path incorporates a large drop off that is essentially a sharp corner. Introducing a plane to the environment and deleting its top-right vertex gives this desired outline. Then using a bevel function the corner can be set with the offset curvature corresponding to the user's input. Converting this geometry from a mesh to a curve the same bevel radius of the remaining tracks can be applied.

Then all that needs to be introduced to the simulation environment is the three spheres that slide down each subsequent track. Physics properties are applied such as friction coefficients for both the track and sphere. For ideal test conditions, coefficients can be denoted as zero. If the user wishes to perform a simulation more realistic interaction between surfaces the values can be adjusted in the initial input arguments. At this point, the entirety of the developed code is in place to set up the brachistochrone test system within seconds once the script is run. The below figures are the first and last captured frames of a successful simulation. In line with the theory, the red sphere taking the brachistochrone track reached the endpoint in the fastest time. It was then followed by the green sphere that was subject to the steep drop off path, and the blue sphere which had the least amount of distance to travel came in last place.

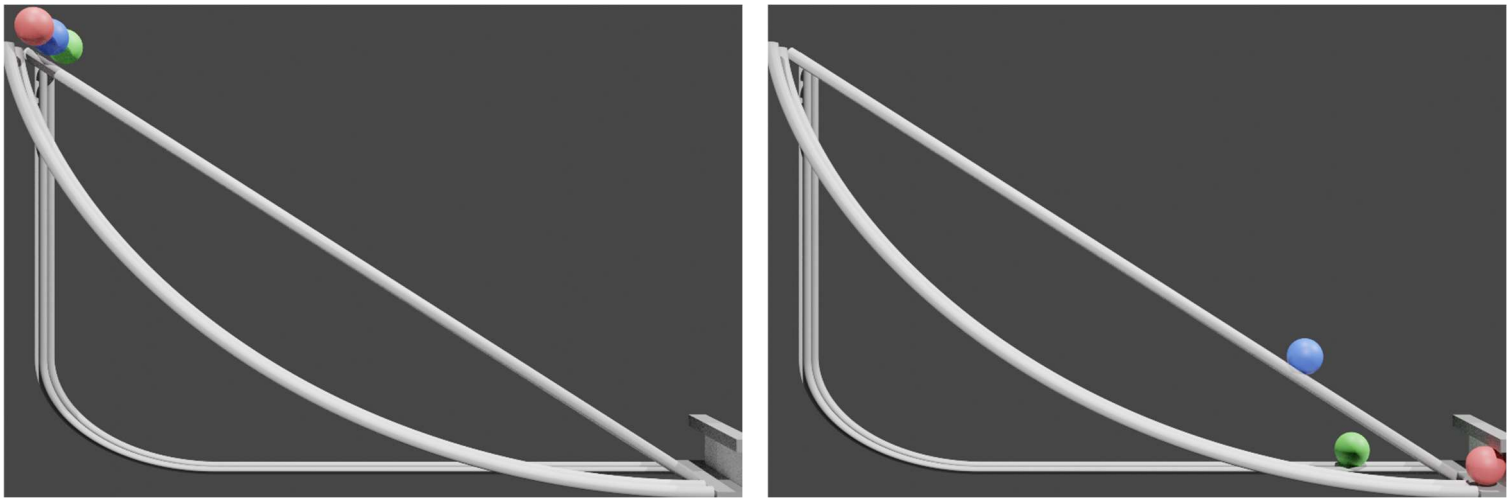


Figure 3.2.3- Conducted simulation first and last frame comparison

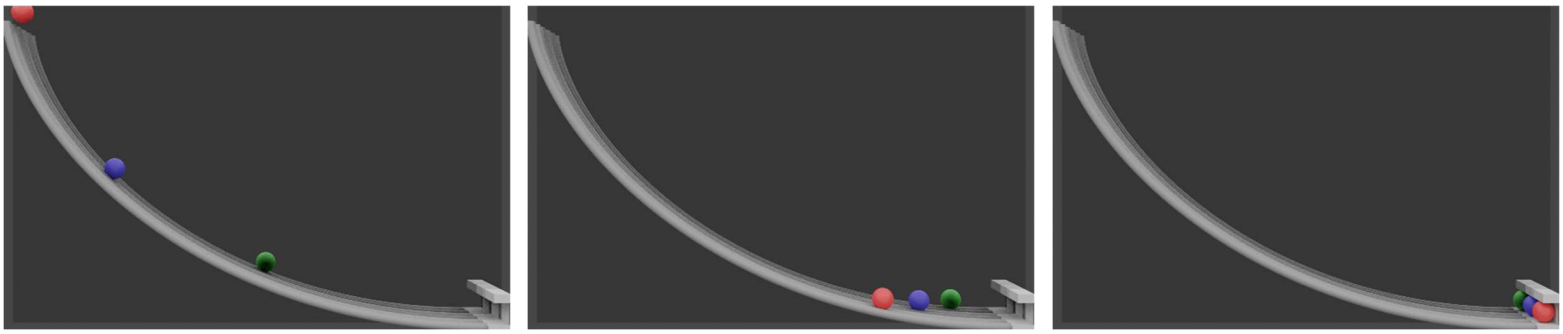
This shows to achieve a fast time the path the relies on how gravity can work in its favour to increase the velocity of the sphere and relies less on the actual distance needed to be covered.

To simulate the curves tautochrone properties the script is quite similar but with a few alterations. Instead of the alternate paths, the script duplicates two further meshes of the cycloid track. Next, the objective was to enable the ability to place a sphere to whatever specific position on the curve which the user intends. Utilizing the visual transformation function seen previously this can be done. The developed code below jumps forward to the frame of users choosing and places the spheres in that specific location.

```
Fb2 = 30  #Set Frame of the second sphere
Fb3 = 45  #Set Frame of the third sphere

bpy.context.scene.frame_set(Fb2) # Moves rigid bodies to respective positions at first set frame

bpy.data.objects['Sphere.001'].select_set(True) # Selects the second sphere
bpy.ops.object.visual_transform_apply(True)   #visual transforms the first mesh to that location
bpy.data.objects['Sphere.001'].select_set(False) # Unselects object

bpy.context.scene.frame_set(0) # Moves back to the beginning frame 0

bpy.context.scene.frame_set(Fb3) #Moves rigid bodies to respective positions at second set frame

bpy.data.objects['Sphere.002'].select_set(True) # Selects the third sphere
bpy.ops.object.visual_transform_apply(True) #visual transforms the second mesh to that location

bpy.context.scene.frame_set(0) # Moves back to the beginning frame 0
```

Blender script screengrab 2.3- Developed code which allows the starting position of spheres to be controlled

Having this capability to control the location gives easy access to a variety in the distance between the spheres for any conducted tests. Below are figures of initial simulations with the spheres spaced evenly apart. As expected, the spheres appear to strike the end of the path at the same moment, backing up the theory explored previously. Watching the video of the simulation is quite satisfying and interesting. To observe it is almost quite similar to a simple harmonic pendulum, where the motion of each sphere is dictated by the same period that of which is $\pi\sqrt{r/g}$.



Frame1 Frame 65 Frame 69

Figure 3.2.4- Tautochrone simulation frame observation

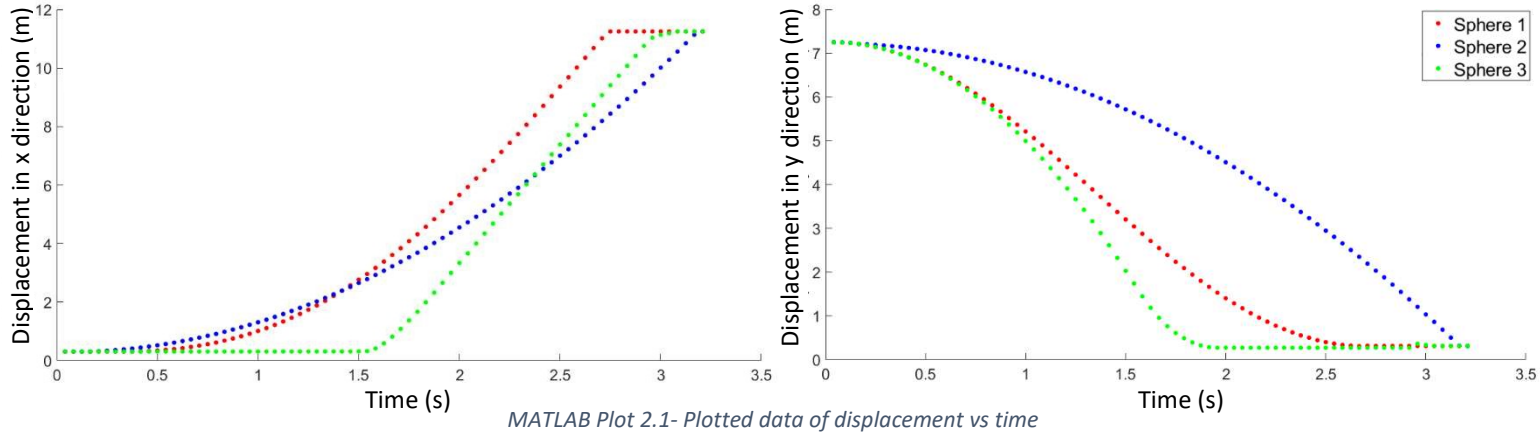
To further investigate the intricacies of both the brachistochrone and tautochrone simulations it was decided to adopt the approach that was taken when tackling the Galton board distributions. This involves exporting the needed data to MATLAB for more versatile analysis. Previously all that was required was the final coordinate of each settled particle within the distribution. In this case, each location the spheres are needed and at each frame of the simulation. Utilizing a similar script that was developed for the Galton board except it loops through each key-frame as opposed to the collection of spheres. At each time step, the code applies a visual transformation to the collection and records the current frame and the x and z coordinates of each sphere. This gives a total of seven rows of data that is exported to a CSV file.

It is then necessary to compile a MATLAB script that takes this information and can calculate characteristics of each body such as velocity and acceleration. Velocity is simply the difference of position with respect to the time step, and acceleration is subsequently the change of velocity within the same period. Having these useful components can give further insight into the mechanics of what is occurring in each simulation.

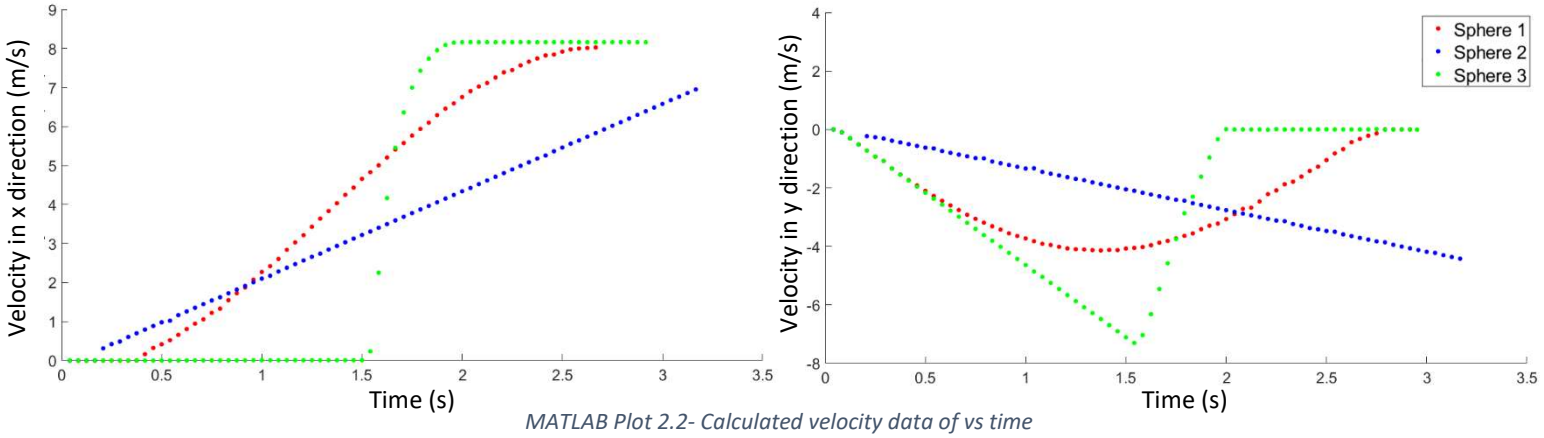
```
fps = 24; % User inputs frame rate to set correct time domain

Array=csvread('bricxz.csv'); % Reads in data csv file
t = Array(:, 1);
x1 = Array(:, 2);
y1 = Array(:, 3);
x2 = Array(:, 4);
y2 = Array(:, 5);
x3 = Array(:, 6);
y3 = Array(:, 7);  % Sets frame and each position data

tn = t/fps;        % Converts time array to seconds

vx1 = diff(x1)/(1/fps);  % Differential of x coordinates equals Vx

vy1 = diff(y1)/(1/fps);  % Differential of y coordinates equals Vy

v1 = hypot(diff(x1), diff(y1))/(1/fps); % Differential of both coordinates gives V

ax1 = diff(vx1)/(1/fps);  %Differential of Vx equals acceleration in x direction

ay1 = diff(vy1)/(1/fps);  %Differential of Vy equals acceleration in y direction

a1 = hypot(diff(vx1), diff(vy1))/(1/fps); % Differential of both velocities gives A
.
.
. % repeats code for the other two spheres data
```

MATLAB script screengrab 2.2- Code which assesses sphere location per frame data

Taking advantage of the graphic capabilities of MATLAB once again these several components can be visualised to give a broader understanding. For the brachistochrone simulation plotting these variables will show just exactly why the path is the route of shortest decent. The object that took the cycloid track is denoted by sphere 1, the linear track is sphere 2, and the vertical drop path object is sphere 3.



MATLAB Plot 2.1- Plotted data of displacement vs time

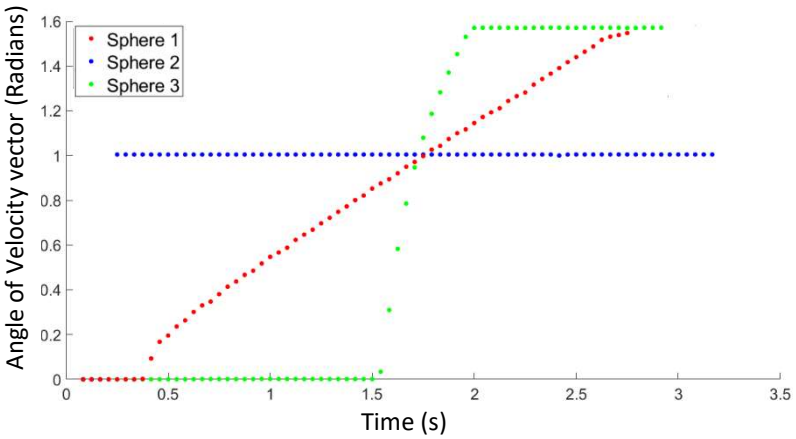


MATLAB Plot 2.2- Calculated velocity data of vs time

As can be seen from the graphs the first two of displacement vs time that the brachistochrone path was the route of fastest descent which was already known. Analysing the velocity of each sphere in the x and y plane it shows that the cycloid curve takes advantage of the only force in play, gravity. It maximises the amount of speed the sphere can obtain due to gravity while doing so in the ideal distance between the start and end points.

Another way to look at this is by reframing the approach to how each sphere location is defined. Instead of describing its trajectory in terms of x and y coordinates, it can be described in terms of the angle the velocity vector makes as a function of time. Which is also the

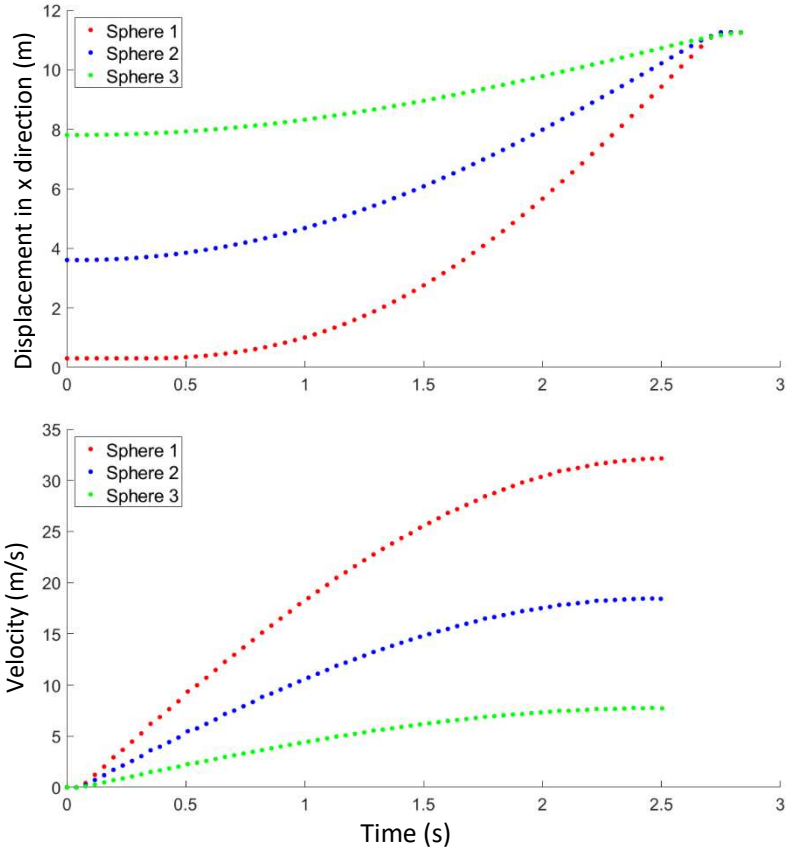
$\tan^{-1}(\frac{v_y}{v_x})$, velocity in the y-direction divided by the velocity in the x-direction. When each sphere is plotted in this t-θ plane, it can be shown that sphere 1 has data that is linear. This is to say θ increases at a constant rate with respect to time.



MATLAB Fig 2.3- Brachistochrone simulation, direction of velocity vector vs time

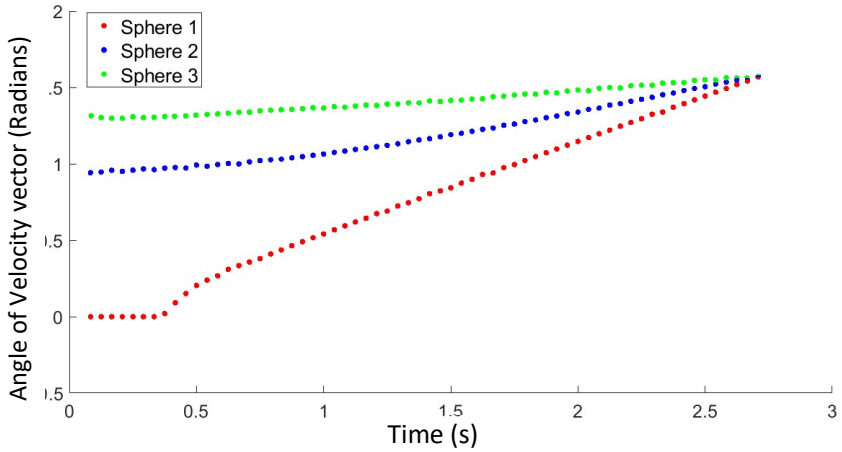
This shows how the brachistochrone path always will be the fastest route, it allows for a constant adapting manipulation of direction and speed that is built into its geometry.

Similarly, the CSV file containing the tautochrone simulation x-y data can be imported into MATLAB so that the mechanics can be truly visualised.



MATLAB Fig 2.4- Displacement in x & overall velocity vs time

The above plots give insight into how each sphere reaches the end point in an equal amount of time. Each velocity profile seems to follow the same type of polynomial function, all peaking at the same moment in time. Although all have different maximum velocity values, each is seen to have the same period across the range from rest to its max speed. This illustrates how sphere 1 has enough velocity to cover the distance between it and the remaining spheres. Similarly, although sphere 3 has significantly less to travel its speed is ideally low for spheres 1 and 2 to catch up within the same time frame. This can be adapted further by once again representing the motion of the spheres in the *t*-θ plane, where θ is the direction of absolute velocity in radians.



MATLAB Fig 2.5- Tautochrone simulation, direction of velocity vector vs time

Apart from a small portion of sphere 1, all plots show theta increasing linearly with time and subsequently converging at the same point. This small segment corresponds to initial moments when it has zero x velocity after being released. However, it is still evident from the graph how each sphere specific velocity direction is in collaboration with the geometry of the cycloid path to cause the decent in an equal amount of time.

For any person that may be interested in viewing videos of any conducted simulations or testing the software for themselves, a selective repository has been uploaded online. Available is a selection of video files as well as all the developed Blender and MATLAB scripts for both the Galton board and the brachistochrone curve. Instructions for managing the software are also given. It can be found on *Github* at the following URL: https://github.com/eoghanp7.

CHAPTER 4: Discussion of Results

Chapter 4.1 Discussion, Conclusion and Suggestions for future possible research:

Throughout the entirety of the previous methodology section, various takeaways can be made. With regards to developments made on the Galton board, it can be said initial progress was slow. But once a level of familiarity operating Blender was reached, much of the developed script was drafted rather swiftly. The main advantage that can be drawn concerning the script is its simple accessibility and overall functionality. Any user can straightforwardly copy and paste the code directly into the text editor and have all the capacities to perform a variety of simulations with inputs of their choosing. That could either fall in the category of academics that wish to improve on existing versions or merely individuals who have an interest in using Blender.

The goal from the beginning as outlined in the project description was to investigate a biased Galton board. Simulations performed on Galton boards of a regular design were a necessary stepping stone to allow for a smooth transition when bias settings were eventually introduced. These initial tests showed it was possible to achieve the standard normal distribution which would be expected from the literature material on the model. The earliest results came with adjusting the size of the scatter pegs radius to show the standard deviation of the final distribution could be altered. This gave a degree of control towards shaping the final distribution, but more was needed. Successive alterations deciding to breakdown the distance between pegs in the x-direction into two variables can be regarded as the foundation for all progressive investigation. When it was apparent these changes caused the final distribution to resemble that of log-normal shape, it was the logical step to test the validity of these findings. Introducing MATLAB was a significant cornerstone which gave the potential to enhance subsequent analysis. The key data and various graphics that MATLAB was able to provide allowed several insights to be gathered. One noteworthy instance was the inspection within the curve fitting tool concluding the outline of sample distribution received an R-squared value

of 0.9262. This first demonstrated that the overall line of thinking was not flawed and there was indeed an imbedded association with a log-normal distribution.

The following analysis representing the data a probability density function also exhibited encouraging results. The fitting representation of the data set did not reach the peaks of certain bins but did give an overall portrayal of the distribution, not just the outline. Determining in the later stages to conduct simulation while increasing the set number of scatter pegs proved to display a more precise fit. Indefinitely increasing this number of rows would seem to refine the distribution but there must be a tipping point where further additions are counterproductive. Future developments could examine this perspective to find an ideal number of rows that can produce the most correlated results. Additional research could also entail searching for a more coherent relationship between the simulation inputs and their weight of impact on the final distribution.

Endeavours exploring the outcomes of the random peg variation showed what was possible on the other side of the spectrum. Discarding these previous key input parameters still produced final distribution attributes one would associate with a Gaussian form. It can be said any related work could add more adaptive capabilities to the system that gave the illusion of randomness but also could be carefully monitored and selected.

With respect to outcomes retrieved investigating the Brachistochrone problem, the key points of interest would lie on how the interties of the system can be unravelled with the inclusion of digital simulations. Having these capabilities again easily accessible by only needing the developed script to conduct tests within seconds. The following MATLAB procedures validated previous theory that was explored while also providing key interactive visualization tools to further grasp the mechanics. Future pedagogical research could hopefully draw inspiration on what was conducted and apply similar approaches to enhance student interaction with content.

In summary, the following aspects can be concluded.

- It was demonstrated that Blender is capable of stimulating physics to a degree of significance for engineering education.
- The Python scripting capabilities allow for simple set up and modification of simulation involving a multitude of different parameters.
- The work conducted here can provide a framework for the development of future dynamical system simulations for use in education.

Lastly, both The Galton board and the Brachistochrone can be looked at for their respective attributes but the combination of the outlook taken on both can serve to benefit a host of dynamical systems. Obtaining the recipe on what is involved to evaluate each model could be similarly applied to other complex systems such as the optimization of packing objects. Overall, it cannot be inaccurate to judge how influential these certain complex arrangements can be in the landscape of overarching engineering principles.

References

[1] Galton Board. (2019, September 9). [cited January 5, 2020], Available from https://galtonboard.com/

[2] O' Connor, J. J., & Robertson, E. F. (2002, February). Brachistochrone problem. [cited February 27, 2020], from http://mathshistory.st-andrews.ac.uk/HistTopics/Brachistochrone.html

[3] Oshman, Christopher, "Experiments with a Galton board" (2002). *Theses*. 705: pp.5-7 [cited January 2, 2020], Available from https://digitalcommons.njit.edu/theses/705

[4] Freauenfeld, M. F. (2018, May 8). Desktop probability machine. [cited January 2, 2020], (Fig 2.1.1), Available from https://boingboing.net/2018/05/08/desktop-probability-machine.html

[5] Wikipedia contributors. (2019c, December 19). De Moivre–Laplace theorem. [cited January 4, 2020], Eq.s(2.1.1, 2.1.2, 2.1.3, 2.1.4), Available from https://en.wikipedia.org/wiki/De_Moivre%E2%80%93Laplace_theorem

[6] Wikipedia contributors. (2020c, January 17). Central limit theorem. [cited January 3, 2020], Available from https://en.wikipedia.org/wiki/Central_limit_theorem

[7] Wikipedia contributors. (2020, January 12). Pascal's triangle. [cited January 16, 2020], Available from https://en.wikipedia.org/wiki/Pascal%27s_triangle#Relation_to_binomial_distribution_and_convolutions

[8] Weisstein, Eric W.("Brachistochrone Problem." From *MathWorld*--A Wolfram Web Resource. [cited March 2, 2020], (Eq.2.2.1-2.2.13) Available from https://mathworld.wolfram.com/BrachistochroneProblem.html

[9] Jaramillo, H., (2015, October 28). How Can I Draw This Cycloid Diagram With Tikz. [online] TeX - LaTeX Stack Exchange. [Cited 4 April 2020], (Fig 3.16), Available from https://tex.stackexchange.com/questions/196957/how-can-i-draw-this-cycloid-diagram-with-tikz

[10] Wikipedia contributors. (2019b, December 15). Tautochrone curve. [cited January 18, 2020], (Fig 3.15), Available from https://en.wikipedia.org/wiki/Tautochrone_curve

[11] Weisstein, Eric W.("Tautochrone Problem." From *MathWorld*--A Wolfram Web Resource. [cited March 4, 2020], (Eq.2.2.14-2.2.21) Available from https://mathworld.wolfram.com/TautochroneProblem.html

[12] Bullet Real-Time Physics Simulation | Home of Bullet and PyBullet: physics simulation for games, visual effects, robotics and reinforcement learning. [Internet]. Pybullet.org. 2020 [cited 14 January 2020]. Available from: https://pybullet.org/wordpress/

[13] He, H., Zheng, J., Sun, Q., & Li, Z. (2019). Simulation of Realistic Particles with Bullet Physics Engine. E3S Web of Conferences, 92, 14004. [cited January 10th 2020], Available from https://doi.org/10.1051/e3sconf/20199214004

[14] Buck Institute for Education. (n.d.). PBLWorks. [cited January 22, 2020], Available from https://www.pblworks.org/what-is-pbl

[15] CAST: About Universal Design for Learning [Internet]. Cast.org. 2020 [cited 19 April 2020]. Available from: http://www.cast.org/our-work/about-udl.html

[16] Irish Universities Association. (2019, November 29). Enhancing Digital Teaching and Learning. [cited January 12, 2020], from https://www.iua.ie/ourwork/learning-teaching/digital-learning/#

[17] Analytica. (n.d.). LogNormal. [cited January 14, 2020], (Fig 3.12), Available from https://wiki.analytica.com/index.php?title=File%3ALogNormal(median%3D3,stddev%3D2).png

[18] Technovation. (2019, July). The Brachistochrone Curve. [cited January 18, 2020], (Fig 3.2.2), Available from https://www.instructables.com/id/The-Brachistochrone-Curve/