Biased Galton Board and Other Exciting Toys

Aidan Lee

School of Mechanical and Materials Engineering

MEEN30180 Professional Engineering Project (Mechanical)

Final Report

Supervisors:
Prof. Vikram Pakrashi
Prof. Kevin Nolan

25 April 2021

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: Aidan Lee

Student Number: 17706551

Report Title: Biased Galton Board and Other Exciting Toys

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:  Date: 24/4/2021

Abstract

Nonlinear dynamics are a subject of growing importance which most undergraduate mechanical engineers know nothing about. The use of flexible structures in engineering is rising and these involve dynamic and nonlinear operation, such as in floating offshore wind turbines which are of great importance to achieve the UN sustainable development goals. As these mechanics are not on the undergraduate engineer's syllabus in UCD and most universities, this project aimed to explore a variety of simple nonlinear and chaotic systems for the purpose of education through simulation. The software, Blender, was used for its intuitive interface and fast deterministic physics engine, Bullet. To investigate Bullet's utility in simulation, an "oscillating Galton board" was simulated, which was compared to a mechanical real-life model. These experiments revealed the shortcomings of Bullet in the simulation of oscillations and collisions. Despite this, Bullet was found to be able to simulate nonlinear chaotic systems with adequate accuracy. When these systems were analysed by the same graphical methods used for real engineering, results were consistent with experimental results from theory and literature. This showed promising potential for Blender's use for pedagogy and digital laboratories to introduce students to nonlinear dynamics and chaos.

Acknowledgements

Firstly, I would like to sincerely thank my supervisors, Professor Vikram Pakrashi and Professor Kevin Nolan, for their generous and valuable guidance throughout my final year project. I greatly appreciate the time taken in teaching me the analysis techniques and giving me constructive feedback on this work.

Special thanks to Ryan Paetzold for sharing his expertise on the laser-cutting process, and to Favour Okosun, Tadhg Buckley and John Gahan for their assistance with my construction of the oscillating Galton board.

Finally, I am extremely grateful to my family for their steadfast support during my time in university.

Contents

List of Figures

1 Introduction

1.1 Motivation

During the COVID-19 pandemic, laboratories were cancelled, and the opportunity for engineering students to undergo enhanced experiential learning of mechanical systems was challenged.

While mechanical experiments are difficult and expensive to conduct, they can be supplemented by digital experiments in a virtual learning environment.

The topic of nonlinear dynamics is becoming increasingly important in various engineering sectors. There is a rise of flexible structures in engineering which have dynamic and nonlinear operational conditions, such as in floating offshore wind turbines or risers and soft robotics. The detection of routes to chaos and instabilities in these systems is crucial to the operation, safety, serviceability and reliability of these systems. However, these fundamental mechanics of nonlinearity and chaos are not included in the undergraduate syllabus of mechanical engineering in UCD and in several other universities global. A core reason behind this is a lack of evidence base and benchmarks that would enable students to have an experiential approach to such concepts.

This project aimed to explore a wide range of nonlinear systems while reducing them to their simplest possible form for teaching purposes. Visualisation of nonlinear phenomena was thus a core aspect of this project.

Learning the dynamics of systems or machines can be difficult when there is only a 2D drawing of the system supplemented with text to aid in the description. Seeing a live demonstration, such as for the phenomenon of precession, can yield a better first-hand learning experience compared to a pre-recorded video. Studies have shown toys to be a powerful cognitive tools for teaching mechanical engineering [7, 8].

Therefore, this project generated simulations of toys that exhibit a range of phenomena, including chaotic behaviour, for the purpose of aiding education through digital experimentation.

Chaotic vibrations are a chronic issue in any dynamic system where there is nonlinearity, and most systems in nature are nonlinear [9]. In order to solve problems in relation to chaos, engineers must be able to identify when it is present and distinguish between vibrations that merely appear to be chaotic. There are various methods to determine if motion is indeed chaotic and to look for routes or precursors to chaos.

Where there is high nonlinearity and determinism, there is chaos. Chaos can be demonstrated with the use of small, simple table-top toys, which this project exploited. These systems were simulated with the animation software Blender, which uses the deterministic physics engine, *Bullet*.

While Blender is normally used for creative work such as films and video games, the software was found to be a powerful learning tool for simulating dynamic systems during this project. Blender can simulate complex systems using an intuitive and and simple interface,

which makes it excellent for students learning dynamics. A traditional dynamical simulation software, such as Ansys, requires complex interfaces to simulate very simple systems. The use of Blender as a dynamical simulation tool is a novel concept and this project aimed to test this idea.

"*Chaotic and Fractal Dynamics*" by Francis Moon [2] delves into chaotic toys as an introduction to chaos and nonlinear dynamics with significant depth and is the only published literature to do so. This book outlined 9 nonlinear chaotic toys and models, 4 of which were chosen to be simulated to detect markers chaos.:

1. Magnetic pendulum
2. Double pendulum
3. Spaceballs toy
4. Rollercoaster chaos model

Another nonlinear dynamic toy is the Galton board which demonstrates the central limit theorem by allowing a large number of small balls to fall through a scattering array of pegs. The balls fall and accumulate in columns at the bottom of the toy, and the distribution is a normal distribution, as asserted by the central limit theorem. Hence, Galton board demonstrates a dynamical system with statistical outputs from stochastic inputs.

In 2020, Eoghan Phelan (UCD [3]) studied the effects of a biased Galton board which had the array of pegs arranged such that the resulting distribution could be controlled. In both the standard and biased Galton board, the pegs are stationary. This project investigated the effects of oscillating pegs on the distribution. In the oscillating Galton board, the pegs oscillate from side to side in the vertical plane.

1.2 Research Objectives

The key "big-picture" objectives for this project were:

1. Develop nonlinear systems in Blender and establish if these systems adequately capture nonlinear phenomena.
2. Investigate chaotic behaviour of nonlinear systems, including markers to chaos.
3. Investigate nonlinearities in dynamical systems with statistical outputs through Blender.
4. Create insights to limits of Blender-driven simulation and contributions to pedagogy.
5. Investigate nonlinear dynamic problems from investigated systems using computation and signal processing.

1.3 Scientific Questions

Research questions were essential such that the objectives could be achieved by answering these questions. Therefore, this project sought out to answer the questions of:

1. What markers of chaos are evident in the simulated nonlinear systems (magnetic pendulum, spaceballs chaos, rollercoaster chaos and double pendulum)?

2. In the oscillating Galton board, how do the distributions change as a function of the frequency and amplitude of the oscillating pegs ?

3. Is there an effect of phase difference between oscillating rows of the oscillating Galton board?

After conducting simulations of the oscillating Galton board, another question arose:

4. Is the observed narrowing a numerical artefact or a physical phenomenon?

1.4 Approach

Simulations for simple table-top toys were created using the animation software Blender for 2 reasons. Firstly, Blender employs an intuitive and attractive user interface that is much more accessible to undergraduate students compared to conventional simulation packages. The second reason Blender was chosen was its implementation of the *Bullet* physics engine, which is very fast computationally.

The results from these simulations were then analysed in Matlab to investigate chaotic behaviour and to establish if Bullet was able to adequately capture nonlinear phenomena. This was conducted using graphical analysis methods (phase planes, Fourier spectra and Poincaré maps) that are used to detect markers of chaos in real systems.

The *oscillating* Galton board was simulated in Blender to investigate nonlinear dynamic problems and nonlinearities in dynamical systems with statistical outputs. The results of the simulations were analysed in Matlab. A mechanical recreation of the digital model was made to compare to the simulation results and to assess Blender's utility in investigation of nonlinear dynamic systems.

1.5 Report Layout

The subsequent sections of this thesis will provide a background to the project's core topics, including a background to Blender and the Bullet physics engine, the use of labs in pedagogy, chaotic theory and an introduction to the Galton board.

The methods to generate the simulated systems in Blender are then explained in moderate detail in section 3 and the graphical methods to analyse the results to detect markers of chaos are outlined. The setup of the mechanical oscillating Galton board is also explained.

In section 4, the markers of chaos in the simulated chaotic systems are investigated and compared to periodic systems. Effects in the digital oscillating Galton board are revealed and compared to the mechanical model to reveal insights to Blender's simulation.

Finally, the results are used as a basis to evaluate Blender as a tool for digital experimentation and pedagogy.

2 Literature Review

Chapter Layout

The literature review in this chapter provides an overview of the importance and benefits of experiential learning through laboratories. The advantages and limitations of digital laboratories in pedagogy are explored in the first section of the literature review.

The review then describes the computational methods used by *Bullet* to achieve its fast yet adequately accurate deterministic simulations and introduces Blender as a digital learning environment.

Nonlinear dynamic and chaos theory are reviewed, with a focus on graphical methods to identify markers of chaos and to distinguish chaos from periodic motion. Sources of nonlinearities and the importance of nonlinearity and chaos in dynamics is discussed in this section.

Finally, the Galton board is introduced as a nonlinear system to be investigated. The design of the Galton board and the central limit theorem are explored.

2.1 Digital Experimentation and Toys in Pedagogy

In the advent of the COVID-19 pandemic, important weaknesses in educational methods were revealed due to the lack of digital pedagogy. On 12 March 2020, the Irish government closed colleges and schools, which led to the cancellation of all the lab sessions in stage 3 mechanical engineering in UCD with no alternatives except additional weighting of written exams and assessments. This was unfortunate due to the lost opportunity to enhance learning of theoretical concepts that laboratories can achieve.

Hofstein suggests that laboratories promote important learning outcomes and leads to intellectual interaction and discussion between students [10]. Furthermore, Feisel and Rosa assert that laboratory objectives can lead to students' more critical judgement of practices in other environments, such as in industry [11].

In the event of a pandemic or natural disaster, traditional laboratories can be substituted with simulations in digital learning. Additionally, digital laboratories enable remote and asynchronous learning and make education more accessible, such as for those that are unable to travel due to location, health, or physical disability. In this way, digitalisation allows academics to share knowledge to a wider audience. Asynchronous learning also enables education method

to be tailored to specific students, which has potential to be further developed by data analysis and personalisation algorithms.

While there are benefits to digital labs over traditional labs, Balakrishnan states that there are considerable drawbacks, including reduced sense of control due to less freedom of configuration, insufficient realism without 3D graphics, and a lack of the interactivity that is present in real labs [12]. However, Balakrishnan maintains that simulations achieve a very close experience and have great advantages in teaching engineering.

Simulations may be used to digitally recreate educational toys, which have been shown to be highly effective in teaching engineering. By reflecting real-world engineering constraints while integrating design with engineering knowledge, these toys exploit the same educational principles as laboratories.

In 1997, a survey in Pennsylvania State University showed that toy models were highly beneficial in understanding the material of a structural engineering course [8]. A study by Phillips et al. [8] found that students showed far better understanding on traditional examination compared to before toys were used for the same school and course.

The literature suggests that toys would be an easy and effective way of introducing engineering students to the concepts of nonlinear dynamics and chaos [7]. These methods of toys and digital learning can be applied in the UCD courses, MEEN20030 Applied Dynamics I, MEEN30010 Applied Dynamics II and/or MEEN41040 Advanced Vibrations, as part of a lab to demonstrate nonlinear systems and how chaotic markers may be identified.

In industry, companies such as Rolls-Royce are transitioning from traditional engineering to "digital-first" methods and approaches to education may also need to evolve in this direction. Moreover, digital and remote learning will reduce demand for transport, thus also reducing carbon emissions and energy consumption.

2.2 Deterministic Simulation in Video Game Physics Engines

Figure 1: *Horizon Zero Dawn*, a video game using the deterministic physics engine, Havok

The animation and video game industries have created a need for physics simulation that is both fast and reasonably accurate. To compute these simulations, video games implement separate pieces of software called physics engines. With these engines, such as Bullet, Havok or PhysX, there is always a compromise between accuracy and computational speed. Due to the high computational speed, these engines are able to simulate and generate results in real-time, which suggests their potential use for digital laboratories.

Blender is a software application that is generally used for creating animations for films or video games. While its use as a tool for investigation of physical and dynamic phenomena is not widely adopted, Blender could be suitable for scientific dynamical simulation as its documentation states that the physics engine is "completely deterministic" [13]. Therefore, there are no random inputs in the physics. With a graphical user interface (Figure 2) and intuitive layout of parameter settings, Blender is well suited as a virtual learning environment relative to more thorough simulation packages such as Ansys.

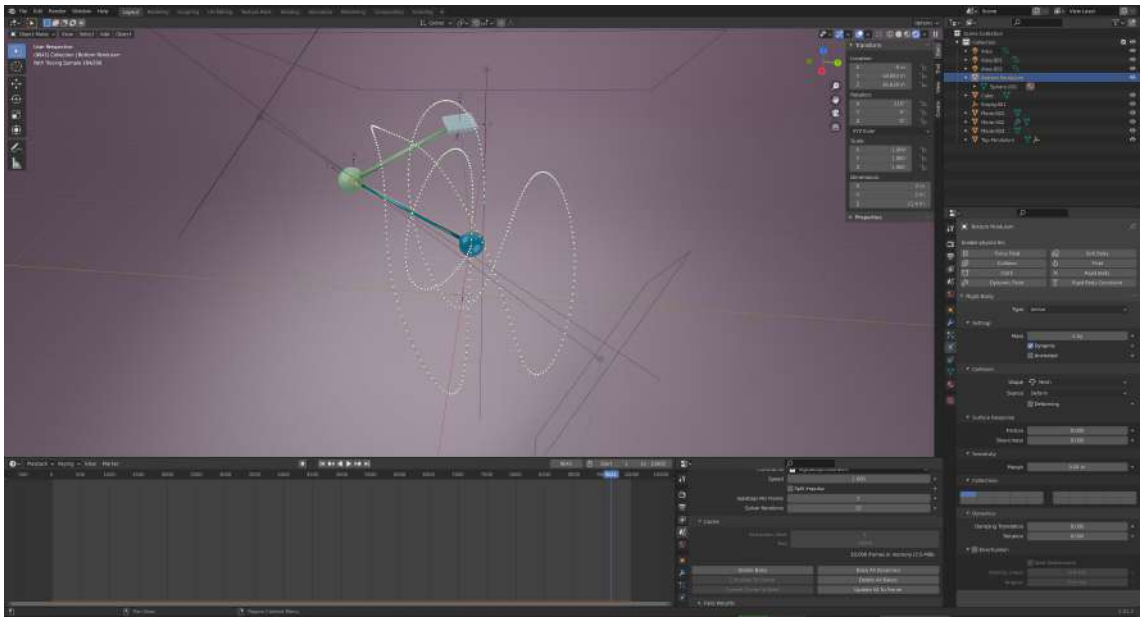


Figure 2: Blender user interface showing timeline, plotted motion path and rigid body physics settings

The Bullet physics engine is implemented in Blender to conduct physics simulation. This engine uses Newton-Euler laws of motion to compute the positions of bodies and the forces exerted on them [14, 15].

A study by He et al. [14] demonstrated the adequate accuracy of rigid body particle simulation with Bullet. Furthermore, the Bullet engine is significantly less computationally demanding than more robust conventional simulation engines. He et al. stated that Bullet was 500-1000 times faster than DEM (discrete element method) code for geotechnical engineering. However, Bullet reduces computational load by assuming constant contact force over the contact duration.

The user interface of rigid body simulation in Blender is much simpler than those of software such as Ansys. For example, Blender does not require specific mesh sizes or parameters to be set before simulation can be started.

Through the use of Python scripts, digital experiments can be conducted automatically to vary some independent variable, such as forcing frequency to find the resonance frequency of a system. Moreover, for dynamical simulations, a motion path can be generated and visualised within Blender (Figure 2), a feature that is not feasible for mechanical laboratories.

A drawback of digital laboratories stated by Balakrishnan [12] was insufficient realism without 3D graphics and a lack of interactivity that is present in real labs. Blender may be able to solve this problem as highly realistic surfaces can be applied its physics objects and rendered in real-time without detriment to performance using the Eevee rendering engine. Furthermore, interactivity is encouraged in Blender as physics variables such as mass and friction can easily be manipulated in Blender, more easily than in a mechanical experiment.

Recently, Blender has officially implemented virtual reality (VR) features that allow users

to enter their scene using VR goggles. The implementation of VR is still unpolished and currently only allows the user to view the scenes and not to interact with the objects. When VR has been developed further in Blender, it could greatly improve interactivity and will open up huge potential for digital laboratories in the future.

Blender is also well suited as a learning environment as it does not require many specific simulation parameters to be set. While this may lead to less accurate simulations, it encourages interactivity, as new physics objects can be easily and quickly placed in the simulated environment. However, while the computational speed of Bullet is excellent, the real-time feedback in response to a variable change is still much slower than a mechanical system.

Blender has great potential in digital pedagogy as it encourages interaction and experimentation with an adequately accurate physics engine that is not computationally demanding, while incorporating an interface that is easy for students to learn. To implement Blender and Bullet in digital laboratories for nonlinear and chaotic dynamics, their accuracy in simulating these dynamics must first be investigated.

2.3 Nonlinearity and the Detection of Chaos

The UN Sustainable Development Goals for 2030 include the objectives of clean energy and sustainable cities, which will require significant expansion of renewable energy capacity. A key technology to achieve this objective is floating offshore wind turbines, which gain access to stronger winds further from the shore. However, these floating platforms lead to high nonlinearities in the system.

Today, additive manufacturing plays a vital role in the progression of the 4th industrial revolution, also known as Industry 4.0. The materials used in additive manufacturing processes are strongly nonlinear and hence involve high uncertainty. Moreover, nonlinearly elastic materials are being used in the development of novel soft robotics to create more flexible and versatile machines.

In an environment that is becoming increasingly nonlinear, the knowledge of engineers needs to adapt in order to solve these novel problems. A particular problem that arises from nonlinearity is chaotic motion, which is only possible in nonlinear systems [16]. When there is chaos, the state of the system cannot be accurately predicted far into the future, even if all the initial conditions and equations of motion are known [2, 17]. Chaos is difficult to identify as it can be easily mistaken for random or quasiperiodic motion. For engineers to be able to solve problems associated with chaotic motion, they must first be able to detect chaos. However, as it is nearly impossible to directly identify chaos, methods to detect routes and markers of chaos are used instead [2].

Most natural systems have nonlinear elements [9], and even the most simple and common electronic circuit elements are nonlinear, such as resistors, inductors, capacitors, transistors and elements involving magnetic induction [18]. In mechanical systems, nonlinearity can be caused

nonlinear spring materials, such as rubber, and the presence of multiple static equilibrium positions that may be introduced by play in joints [2]. These nonlinearities create complications as the principal of superposition cannot be applied and resulting oscillation frequency may depend on the forcing amplitude rather than forcing period [19, 20].

Chaos may be defined as motion in deterministic with systems with a time history that is highly dependent on initial conditions [2, 21]. There is no rigorous definition for chaos that is not limited to only very specific mathematical problems. This is due to the continuous discoveries of new complexities in nonlinear mechanics [2, 22]. Hence, proposed definitions try to use descriptions of unpredictability in the system progression.

The unpredictable nature of chaos has led to the term being incorrectly conflated with "random". In random systems, input parameters such as force are unknown and only some statistics and probabilities of the system outcome are known. In other words, random systems are not deterministic. Chaotic systems are deterministic by definition and as such, have no random or unpredictable inputs [2, 17].

While the random input of noise is present in both experimental and simulated data, it is assumed that large nonperiodic signals do not result from input noise of small amplitude [23]. Therefore, in order to attribute a nonperiodic response to a deterministic behaviour in the system, a large ratio of output signal to noise is required.

Moon [2] categorised nonlinear deterministic motion into 7 classes:

1. Predictable regular motion
2. Unpredictable regular motion
    - The long-time motion is sensitive to the initial conditions, as it is in chaotic motion.
3. Transient chaos
    - This class of motion is also described by Lai and Tél [24].
    - The transient chaotic motions eventually settle into regular motion.
    - The strange attractor is indicated by fractal-like properties in the Poincaré map or phase portrait (Figure 12).
    - Moon suggests conducting an experiment over a long length of time to find transient chaos, even if the Poincaré map appears to map out a fractal structure.
4. Intermittent chaos
    - There are nonperiodic bursts of irregular motion (Figure 7).
5. Limited chaos
    - Chaotic motion occurs with a phase space which remains close to some periodic orbit.

- The frequency spectra often show narrowing or broadening of certain frequency spikes.

6. Weak large-scale chaos

   - Spectra exhibit broad frequency range below the driving frequency, if present (Figure 3).

7. Strong large-scale chaos

   - occurs where there are many degrees of freedom.

5 out of these 7 classes of deterministic nonlinear motion are chaotic. Moon [2] suggested 7 properties to be used to characterise chaotic motion, many of which are also suggested by supporting literature. The accompanying figures are sourced from [2] to provide context and an interpretation to the chaotic phenomena.

1. Generation of a continuous spectrum of frequencies below a single input frequency [2, 25, 26]. (Figure 3)

2. Loss of information to indicate initial conditions [2, 27]. (Figure 4)

3. Uncertainty growth with time [2, 28, 29]. (Figure 4)

4. Broad spectrum of the Fourier transform when the motion is generated by a single frequency [2, 30]. (Figure 5)

5. Fractal properties in the phase space (Figure 6). This property indicates a strange attractor and can be measured by Poincaré maps and fractal dimensions. (Figure 6)

6. Complexity of regular motions increases as some parameter is changed [2, 31]. For example, fluid flow becomes chaotic when velocity is increased to be high enough for turbulence to occur.

7. Intermittent/transient chaotic motions (Figure 7) [2, 24]. These are bursts of randomlike motion that settle into regular motion. This can be observed in fluid flow that is near the transition to turbulence.

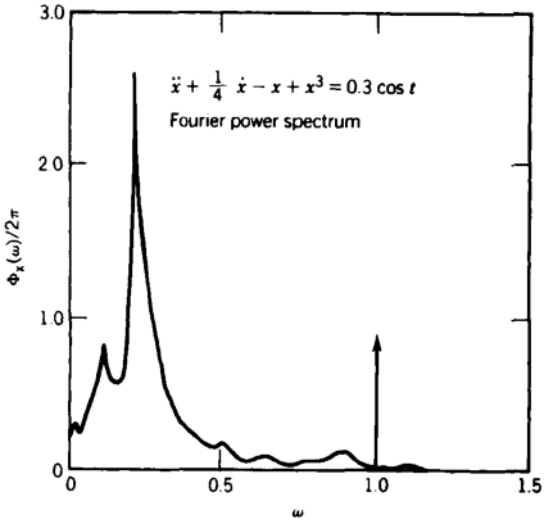


Figure 3: Fourier power spectrum of chaotic motion: Power plotted against angular frequency

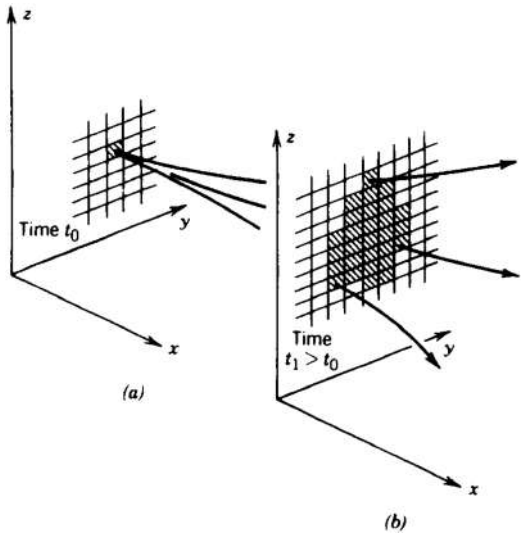


Figure 4: (a) Illustrates the initial conditions and initial uncertainty, represented by the shaded region. (b) As time progresses, the shaded region expands and information is lost about the initial condition.

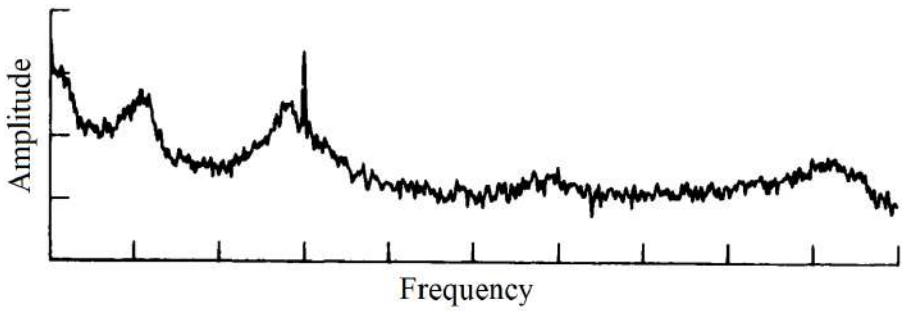


Figure 5: Frequency spectrum - broad band response of buckled elastic beam due to chaotic vibration caused by large excitation

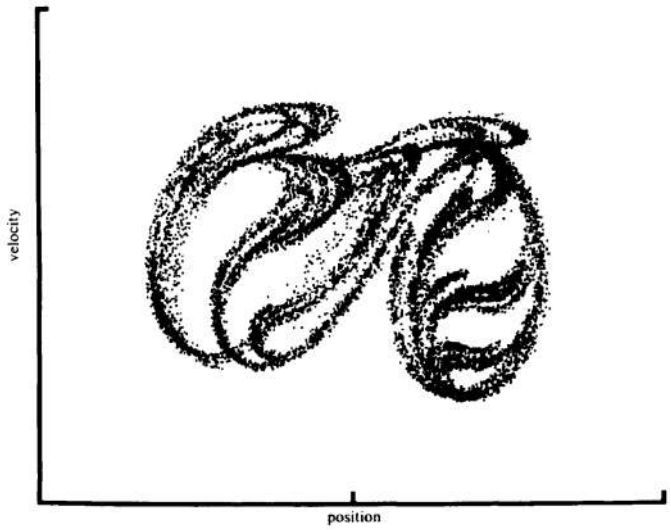


Figure 6: Double Poincaré map showing fractal structure characteristics of a strange attractor

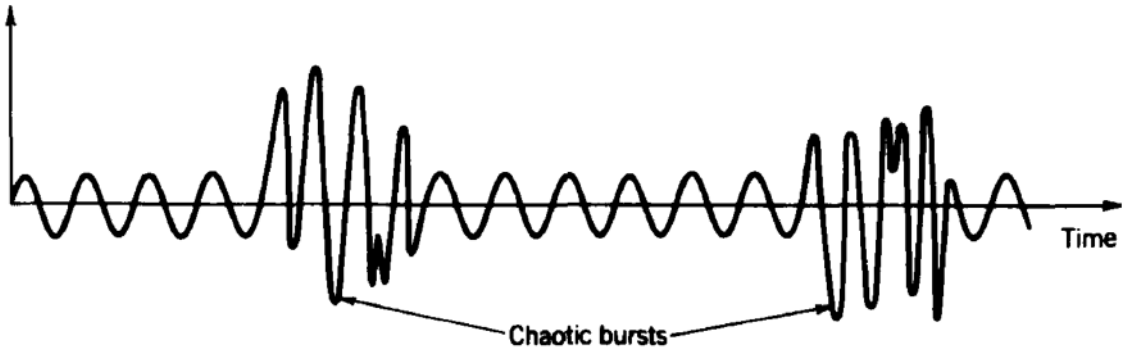


Figure 7: Intermittent chaotic motion

Quasiperiodic motion is also often mistaken for chaotic motion, as it appears to be unpredictable before closer inspection. Quasiperiodicity occurs when there are multiple incommensurate periodic signals in a nonlinear system [2, 32].

Moon [2] suggests 2 methods to distinguish between chaotic vibrations and quasiperiodicity caused by 2 incommensurate signals. Firstly, the Fourier spectrum can be used. If the Fourier spectrum is broad and continuous it may be chaotic. However, if the Fourier spectrum contains 2 spikes, the solution is quasiperiodic and not chaotic [2, 33]. Borkowski and Stefanski [33] state this method is well-established and commonly used to to investigate and diagnose dynamic systems. The other method to differentiate between chaotic and quasiperiodic motion is the Poincaré map, which will be described later in this thesis.

To distinguish between chaotic and nonperiodic or periodic motion, one can use a checklist of 7 qualitative properties:

1. Identify what nonlinear elements are present in the system [2].

2. Check if there are sources of random input [2].

3. Observe the time history of the measured signal [2, 5].

4. Look at the phase planes [2, 5].

5. Examine the Fourier spectrum of a signal [2, 5].

6. Create a Poincaré map [2, 5].

7. Vary system parameters to look for bifurcations and routes to chaos [2, 5].

Moon [2] suggests that at least 2 of these methods should be used to determine if a system is chaotic.

In addition to these qualitative methods, Moon [2], Özer and Akin [5] suggest investigation of a quantitative measure of chaos, the Lyapunov exponent. Moon and Theiler [34] also propose the use of the fractal dimension as a quantification of chaos. Özer and Akin assert that these quantitative and qualitative methods are in full agreement.

In the time history of signal amplitude graphed against time, chaotic vibration can be identified as it will display no visible pattern or periodicity on the chart [2, 5]. However, if the motion has long-period behaviour, there may be no visible pattern or periodicity despite the motion being non-chaotic. Furthermore, the aforementioned quasiperiodic vibrations are also easily mistaken for chaotic using the time-history method. Özer and Akin [5] suggest that this method is the easiest for detecting chaos.

The periodicity, or predictability, of the time history can be measured using the autocorrelation function, $A(\tau)$, given by equation 1 [2, 35]:

$$A(\tau) = \frac{1}{T}\int_{T}^{-T} f(t)f(t+\tau)dt \tag{1}$$

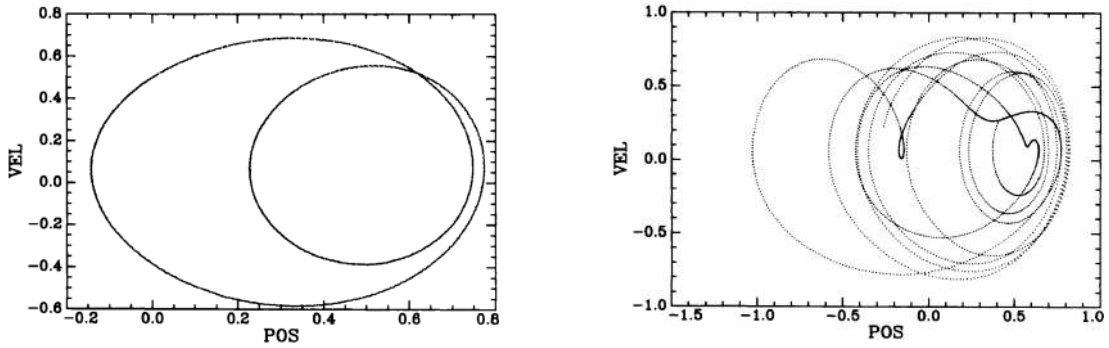
This is a function of the chosen value of time delay, τ. The value is chosen such that it is very large relative to the dominant periods of the motion of interest. For chaotic or random signals, $A(\tau)$ goes to 0 as τ approaches the characteristic time of the motion [2, 35]. The characteristic time, τ_c, is a measure of the time that the motion can be predicted into the future. This function can be used to calculate the periodicity of the signal [36].

A phase space is a multidimensional space in which each dimension is a state variable, such as position and velocity. On a phase plane, periodic vibration appears as a closed curve, as shown by Figure 8a [2, 5]. For a curve to be considered "closed", it must end where it started and be continuous throughout its length. A closed curve can appear for both linear and nonlinear systems, however, only the nonlinear system can display a curve that crosses itself in the phase plane. Moon [2] states that if this type of orbit is observed, then the oscillation is subharmonic, and the forcing frequency is greater than the resonance frequency.

Autonomous systems are defined as systems for which the forcing does not depend explicitly on time [37], and can also be nonlinear. Despite having no periodic forcing, autonomous

systems can result in periodic motions, referred to as limit cycles, which appear as closed curves on the phase plane [38].

Chaotic oscillations do not display repeating orbits on the phase plane, as shown by Figure 8b [2, 5]. Additionally, chaotic orbits are likely to fill a section in the phase plane, as shown by Figure 8b. While nonlinear phase planes can display closed curves (wherein the start and end of the curve are the same), chaotic motions have phase plane trajectories that never close. However, experimental results from Zaher [1] showed a phase plane of chaotic motion with a closed curve (Figure 9). Özer and Akin [5] assert that on the phase plane, chaotic motion will display distinct shapes. While these observations are indicators of chaos, to properly identify that the motion is chaotic, Moon [2] contends that a Poincaré map is required.



(a) Phase Plane of periodic and forced motion of a buckled beam [2]

(b) Phase Plane of chaotic and forced motion of a buckled beam [2]

Figure 8: Periodic and chaotic motion on the phase plane

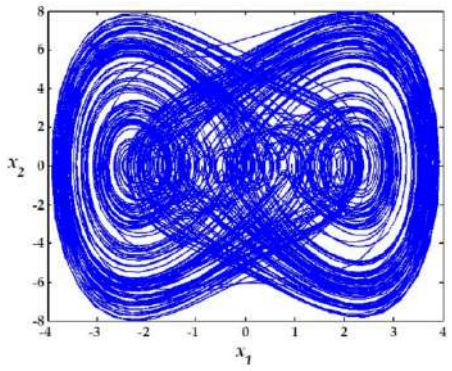


Figure 9: Phase plane of a chaotic duffing oscillator showing closed curves [1]

In 1 degree-of-freedom (DOF) systems, a "pseudo-phase space" can be used to identify chaotic motion [2, 39, 40]. In these 1 DOF system, the signal can be plotted against the subsequent value, after a fixed time delay constant, T. In other words, the signal, $x(t)$, is plotted against $x(t+T)$. The chosen value of T is not crucial, except to avoid choosing the natural period of the system for which resonance would occur. For systems with 2 state variables (2 DOF), Moon [2] suggests using multiple values of time delays, such as using T and $2T$. This constructs a 3-axis space with $x(t)$, $x(t+T)$ and $x(t+2T)$ on each axis. The pseudo-phase

space method is founded upon the concept that $x(t+T)$ is related to $\dot{x}$, the rate of change of x with time.

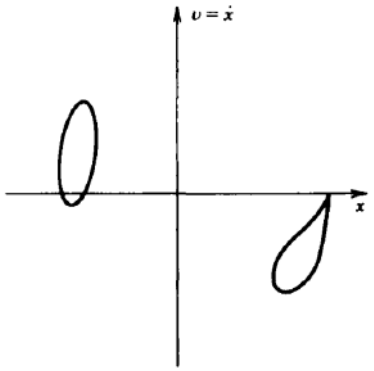
As described previously, the Fourier spectrum of a signal may be used to distinguish between chaos and quasiperiodicity, as a chaotic signal would display a broad continuous spectrum whereas a quasiperiodic signal would display spikes at its incommensurate forcing frequencies. In the chaotic Fourier spectrum, the broad continuous spectrum occurs when its input is a single frequency harmonic motion [2, 5, 33]. Moon [2] stresses that this characteristic of chaotic motion is especially important for systems of low dimension with only 1-3 degrees of freedom. The continuous frequency spectrum output alone does not indicate chaos, as a system can have as many resonant frequencies as it has degrees of freedom. Hence, if the system has many hidden degrees of freedom, it will have a multibaryonic output and can be mistaken as chaotic. Therefore, Moon highlights that the Fourier spectrum is useful to detect chaotic vibrations only when there are observed changes in the spectrum as some parameter is varied.

Furthermore, scaling properties of the Fourier spectrum of the time series can be used to distinguish between chaotic and strange-nonchaotic attractors [2, 5]. In this method, one of the system's forcing frequencies is sampled and its Fourier transform is defined, $|S(\omega)|$. The spectral distribution function, $N(s)$, is defined as the number of peaks of $|S(\omega)|$ which have an amplitude greater than s. For strange-nonchaotic motions, N is equivalent to s^{-a}, where $1 < a < 2$. Therefore, a strange-nonchaotic attractor can be identified and distinguished from a chaotic attractor.

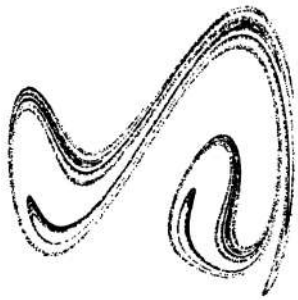
The Poincaré map is a modified phase plane method and a powerful tool to identify chaotic motion [2, 5, 41]. In the phase space, instead of plotting deterministic motion deterministically, the deterministic motion is plotted in discrete time steps, t_n, such that periodic motion appears as a sequence of closed dots in the phase plane. Moon [2] states there there are a certain set of rules to follow for the selection of t_n to produce a Poincaré map.

Moon and Holmes [42] have used a sampling frequency synchronous with forcing frequency to produce a Poincaré section for a dynamic system diagnosis. This allows periodic and nonperiodic functions to be distinguished, where these motions can result from periodic forcing. Using this method, quasiperiodic motions can be distinguished from chaotic motions. The Poincaré map produced in a phase plane would form a continuous closed figure if the motion were quasiperiodic (Figure 10a), thus identifying that the motion is not chaotic [2, 5]. When sampling a periodic system at a rate equal to the a forcing frequency, a finite number of small clusters may appear on the Poincaré map [6].

For autonomous systems, which do not have an associated forcing frequency, a Poincaré map may be constructed by using a 2D plane to "cut" the trajectory of motion in the 3D phase space, and sampling the points of intersection where the trajectory "enters" the plane, as shown by Figure 11 [43, 2]. This method also works for periodically forced systems [43].



(a) Phase Plane Poincaré map of quasiperiodic motion of periodically forced beam in a two-well potential [2]



(b) Poincaré map of chaotic motion of a two-well potential oscillator [2]

Figure 10: Periodic and chaotic motion on the phase plane

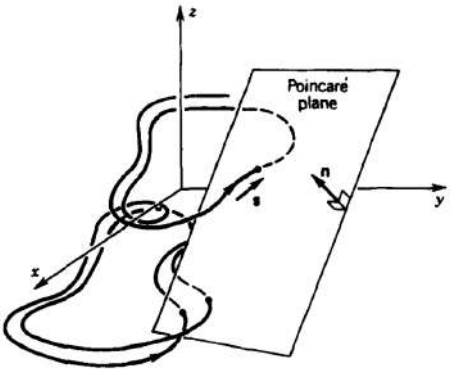


Figure 11: Cutting of phase space trajectory using a plane to form a Poincaré map [2]

For conservative, undamped or negligibly damped systems that are chaotic, the Poincaré maps often appear as a cloud of unorganised points in the phase plane, shown by Figure 13. These are termed "stochastic" motions [2]. In damped, or dissipative, chaotic motions, the Poincaré map can sometimes appear as an infinite set of highly organised points [2, 5]. As one zooms into the infinite set, a structure will appear. If the structure remains as one continues to zoom into it, then the map is termed "fractal-like", which is strongly indicative of chaos and a "strange-attractor" (Figure 6, Figure 12,) [2, 44]. This fractal-like structure is an embedding of structure within structure and this is termed a "Cantor set".

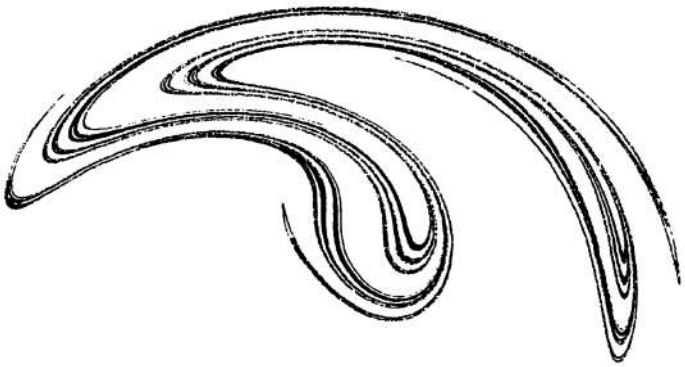


Figure 12: Strange attractors for a periodically forced Duffing oscillator for a circuit with a nonlinear inductor [2]

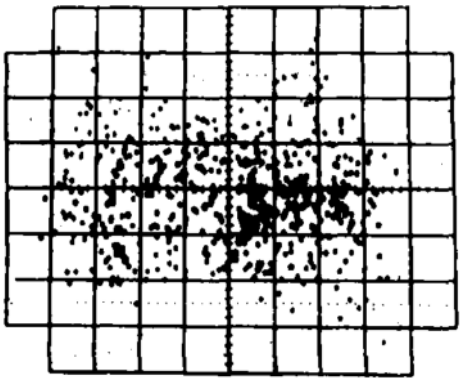


Figure 13: Poincaré map of chaotic motion of a buckled beam with low damping [2]

In 1982, Benoît Mandelbrot, who created the term "fractal" in mathematics, gave a quantitative definition of the fractal as a "set for which the Hausdorff-Besicovitch dimension strictly exceeds the topological dimension". 4 years later, in 1986, Mandelbrot re-defined the fractal in qualitative terms, as "a shape made of parts similar to the whole in some way" [17].

Moon [2] argues for the usefulness of fractal structure in identifying chaos in dissipative systems. Brogliato et al. [45] states that the key feature of dissipative systems is that "the total energy stored in the system decreases with time".

Another map that can be used to identify chaotic motion is the return map [2, 46]. This maps the signal, x_n, against its successor, x_{n+1}. The system can be deemed chaotic if 2 criteria are meet. Firstly, the points, x_n and x_{n+1} must appear to be grouped with some functional relation. Secondly, the function $f(x)$, where $f(x_n) = x_{n+1}$, may be multivalued for a chaotic system [2, 47]. If this function is the output response of a system with significant dissipation, Moon [2] asserts that a polynomial mapping may be used for numerical experiment or analysis.

The literature describes 3 routes to chaos:

1. Bifurcations

2. Quasiperiodicity

3. Intermittency

Routes to chaos in the response of a system can be found by changing control parameters, where through the parameter changes, there may be transition from periodic to chaotic and back to periodic motion [2, 48]. These control parameters could be parameters such as forcing amplitude or forcing frequency. Changing the control parameters can allow one to observe if the system has steady or periodic behaviour for any range of the parameters. Chaotic systems can exhibit subharmonic periodic vibrations as some parameter is varied, which is a characteristic precursor to chaotic motion [2, 49].

Bifurcation is a route to chaos during which the phenomenon of period-doubling occurs. In this phenomenon, an initially periodic system undergoes bifurcation and the motion becomes periodic with double the period of the initial oscillation as some parameter λ is varied. Bifurcation occurs again when λ is varied further, to double the period of the second period (i.e. $1/4$ of the initial period) [50]. Period doubling bifurcation is especially possible in systems with significant dissipation. The values of λ for which bifurcation occurs are given by equation 2 [2, 51]:

$$\frac{\lambda_n - \lambda_{n-1}}{\lambda_{n+1} - \lambda_n} \rightarrow \delta = 4.6692016 \tag{2}$$

Moon [2] stresses that generally in practice, the limit approaches δ after the third or fourth bifurcation. The bifurcations continue to occur until a critical value of λ, after which the motion becomes chaotic, and the chaotic motion may exit in a finite band of λ values [2, 50]. In this way, bifurcations are a route to chaos. The phenomenon of 'crisis' may occur wherein the chaos suddenly disappears when the bifurcation parameter reaches a certain value [50]. A Poincaré map may be used to easily observe period doubling and subharmonic bifurcations by sampling a dynamic variable [2, 48].

Bifurcation diagrams are a technique to examine prechaotic and post chaotic changes under λ variation [2, 50]. The bifurcation diagram plots motion as a function of a system bifurcation parameter. The motion is measured by a response variable such as maximum amplitude. If the bifurcation diagram loses continuity after some value of λ, then the system could have gone into quasiperiodic or chaotic motion. The logistic map (Figure 14) shows period doubling where the line splits (bifurcates), followed by transition into chaos as λ is increased further.

Quasiperiodicity, while being periodic and often mistaken for chaos, is a route to chaos nonetheless [2, 50, 52]. This refers to the precursor to chaotic motion in a periodically forced system with multiple simultaneous periodic oscillations each with a frequency which are incommensurate.

Intermittency (Figure 7)is a route to chaos that involves bursts of chaos where there are long durations of periodic motion [2, 53]. These chaotic bursts become more frequent and occur over a longer duration once a particular parameter is varied.

The literature described thus far has been concerning qualitative methods of identifying markers and routes to chaos. Quantitative measures of chaos are Lyapunov exponents and

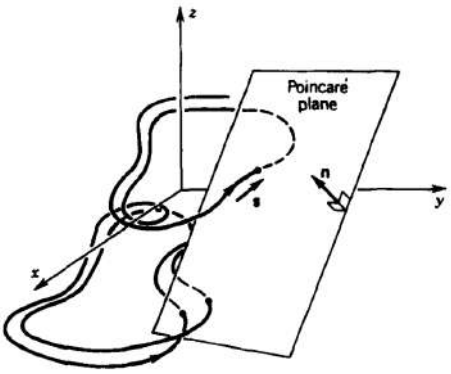


Figure 14: Period-doubling bifurcations for the logistic map. [2]

fractal dimensions.

The Lyapunov exponent is a measure of the growth of uncertainty with time. When it is positive, it indicates chaotic dynamics due to the exponential growth of uncertainty with time [2, 54].

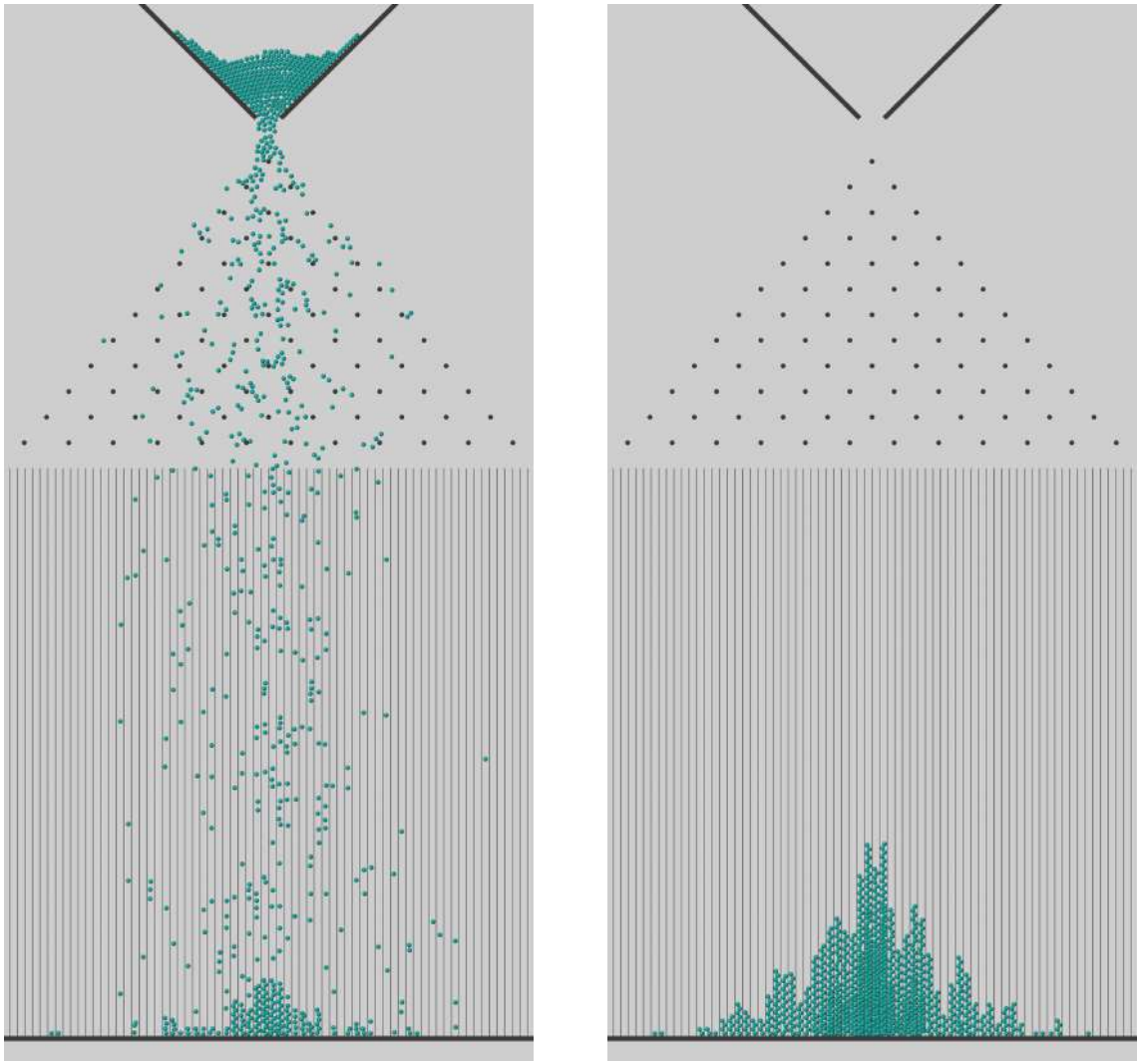
Wolf et al. [54] suggest various algorithms to quantify chaos by estimating Lyapunov exponents, which require large quantities of accurate data and an attractor of not very high dimension. Moon [2] and Wolf et al. both maintain that there is a lack of robust methods to calculate the Lyapunov exponent.

Feder [17] provides a method to calculate the Hausdorff-Besicovitch and fractal dimension, D. This involves counting the number of spheres required to cover a set of points, and by doing so, the size of the set is being measured. The length of the curve must be found by finding the number of line segments, N, needed to cover the line. An area is then associated with the set of points defining the curve by giving the number of disks, equal to N, needed to cover the curve. A volume, V, is associated with the line. The line segments are given length δ, which is equal to the diameter given to the spheres. By choosing different values of δ, N can be expressed as a function of δ. The fractal dimension, D, can then be determined by the slope of $\ln N(\delta)$ plotted against $\ln \delta$. This definition is described by Moon as the most basic definition of the fractal dimension.

While quantitative tests for the identification of chaos can be automated by computer programs, Moon [2] maintains that human experience and judgement are still essential to provide an accurate assessment as to whether a given motion is chaotic or a strange attractor.

In order to solve problems caused by chaotic motion in nonlinear systems, engineers need to be able to identify and quantify chaos. Many methods described by the literature involve visual properties in plots, which will be helpful to the learning of undergraduate engineers. As visual learning has been argued to be an effective method for teaching science and engineering, these visual methods (e.g. Poincaré map, Fourier spectra, bifurcation diagrams, etc.) can be used to teach engineers chaos in a laboratory setting [55].

2.4 The Galton Board



(a) Balls falling through standard Galton board

(b) Fallen balls form normal distribution in standard Galton board

Figure 15: Standard Galton board

A Galton board is a nonlinear dynamic toy that demonstrates the central limit theorem. In this model, a series of balls fall under gravity, into an array of pegs. After hitting a peg, each ball has a 50/50 chance of going to the left or right peg below it. After many balls have gone through this array of pegs, the result at the bottom is a normal distribution.

The central limit theorem is a phenomenon in which a stochastic (random and unpredictable) input results in a predictable statistical distribution that can be reliably reproduced in repeated tests. Kwak and Kim [56] define the central limit theorem as the claim that a distribution with mean, μ, and variance, $\frac{\sigma^2}{n}$, will result from a sample of a population of size, n and mean, μ, and variance σ^2, for a sufficiently large sample size in which each sample is independent from each other. In simpler terms, when there is a random distribution of sufficiently large size, it will form a normal distribution (Figure 16).

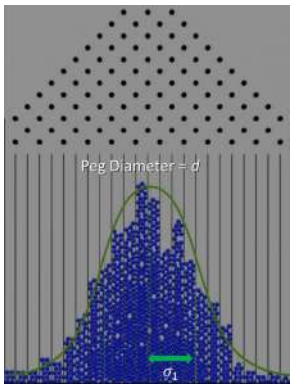


Figure 16: Normal distribution with standard deviation σ_1, formed by a standard Galton board [3]

In the Galton board, each row of pegs represents an independent trial. After the balls fall through the peg array, the central limit theorem asserts that when the balls fall to the bottom, a normal distribution will be formed.

Kozlov and Mitrofanova [57] describe the Galton board as an "upright board with evenly spaced nails driven into its upper half" with a funnel placed directly above a nail in the second row such that a perfectly centred ball will fall directly onto the "uppermost point" of the nail surface. The nails described may also be referred to as pegs.

An alternative design of the Galton board described by Moran et al. [58] has the pegs arranged in a triangular lattice arrangement (Figure 15). This design uses less material and time to manufacture, with little to no detrimental effect, as the balls do not tend to encounter pegs in the top corners of the array.

Kozlov and Mitrofanova maintain that despite imperfections in construction and unavoidable accuracies, when there are sufficiently numerous balls, the resulting distribution at the bottom is a normal distribution due to the central limit theorem.

Phelan [3] was able to control the distribution's standard deviation by altering the peg radius. By varying the horizontal distance between pegs in the peg array, Phelan was able to produce a log-normal distribution with a Galton board.

The Galton board is relevant to this study as it demonstrates a nonlinear system that has predictable control over a stochastic input. The distribution of the balls provides a visual representation of this, and the model can incorporate deterministic dynamic simulation, such as contact dynamics, to achieve this.

2.5 Conclusions

The literature showed that laboratories are crucial to the intellectual development of students. Digital environments can make this experiential learning more accessible and better tailored to individual students. However, reduced configuration freedom in experiments, insufficient realism without 3D graphics and a lack of interactivity have made conventional simulation tools

less suitable for digital labs in education.

Blender is able to tackle some of these problems with its high speed physics engine, *Bullet*. While it makes some sacrifice to accuracy to achieve this, it has been shown to be adequately accurate for simulations in geotechnical engineering. The intuitive environment of Blender and the speed of Bullet create potential for this package to be implemented in pedagogy.

Chaos is a problem that can occur in any dynamic system that involves nonlinearity, which most real systems do. There are numerous methods to identify markers of chaos in the motion of nonlinear systems, many of which are graphical plots, such as Poincaré maps, phase planes and Fourier spectra. These methods can identify nonlinear and chaotic phenomena, and can be used to distinguish between chaotic and periodic motion. The visual nature of the analysis will be beneficial to the students' learning if this content were introduced to them.

The oscillating Galton board is a system that this project simulated to investigate nonlinearities in a dynamic system with statistical outputs through Blender. The standard Galton board visually and dynamically demonstrates predictable control over stochastic inputs.

The elements from each of these chapters can be integrated to investigate nonlinear systems with chaotic behaviour and statistical outputs to create insights to the limits of Blender-driven simulation and contributions to pedagogy.

3 Methodology

3.1 Digital Experimentation with Blender

Blender, normally a software used for film and game animations, was used for its deterministic physics engine and simple interface, which makes it ideal for students learning dynamics. This software implements physics simulation using the Bullet Physics engine, which is an open-source project used primarily in the film and gaming industries. This engine uses Newton-Euler laws of motion to compute the positions of bodies and the forces exerted on them [14, 15]. However, the Bullet engine reduces computational load by making compromises in accuracy.

Objects and bodies in Blender must be imported or created as meshes made up of a finite number of vertices, edges and faces.

3.2 Simulation of Toys to Demonstrate Nonlinearities

Various systems were simulated to demonstrate nonlinearity and chaos for different types of inputs or forcings, including periodic, autonomous and stochastic. In each of the systems, the motion path (position over time) of a particle can be exported from Blender to a CSV file such that it can be subsequently analysed in a suitable application, such as Matlab.

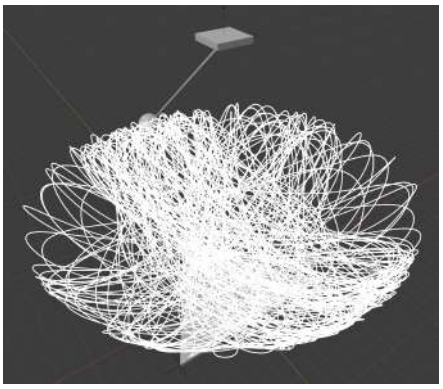


Figure 17: Magnetic Pendulum in Blender with Plotted Motion Path

3.2.1 Chaos in Pendulums

The magnetic pendulum was chosen to demonstrate how chaos arises in an autonomously forced system with forcing independent of time. In this system, the pendulum is forced by 3 repulsive magnets, depicted in Figure 18 as blue icospheres. These magnets apply a repulsive force to the pendulum. As a result, the pendulum is forced nonlinearly as it swings.

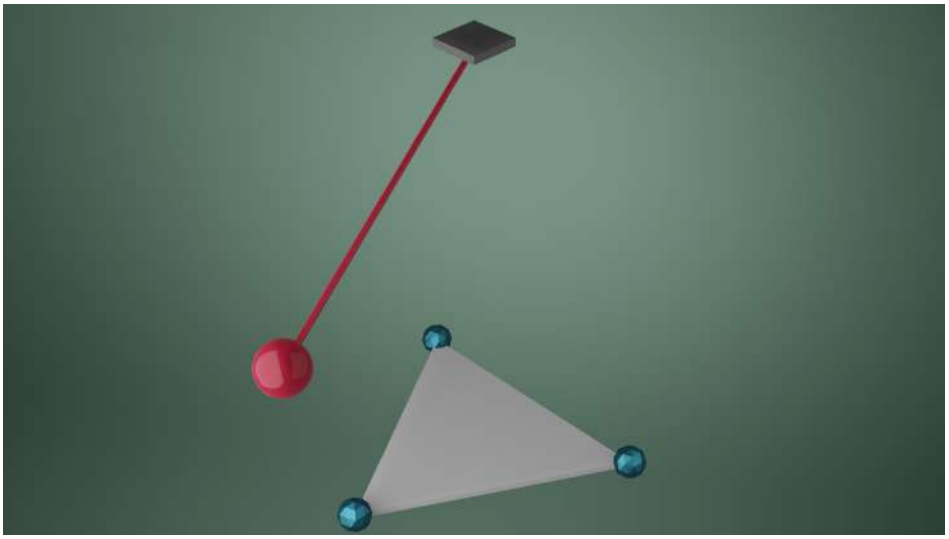


Figure 18: Magnetic Pendulum with repulsive magnets shown as blue icospheres

In order to create the magnetic pendulum dynamic system, a cylinder *mesh* was created with 3 vertices such that a platform in the shape of an equilateral triangle was formed at the base of the system. An *empty* (single-coordinate point) was placed at each of the vertices at the same height (Figure 19). In the *Physics Properties* of each empty, *force field physics* were enabled with type *magnetic* and a defined strength (Figure 20). This makes the empty emit a repulsive magnetic field. A *strength* of 2 was set for the conservative magnetic pendulum system such that the effects of the magnetic fields were clearly visible while avoiding large accelerations of the pendulum.

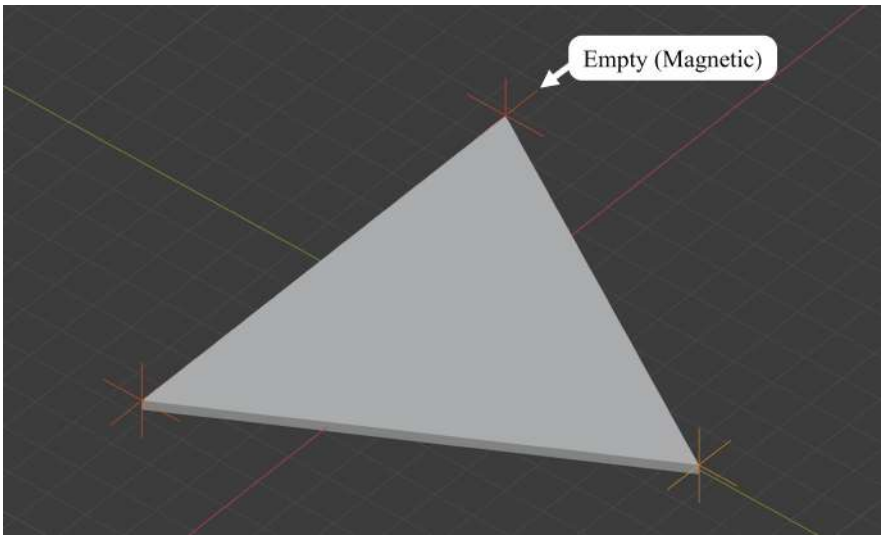


Figure 19: Equilateral base with magnetic empties

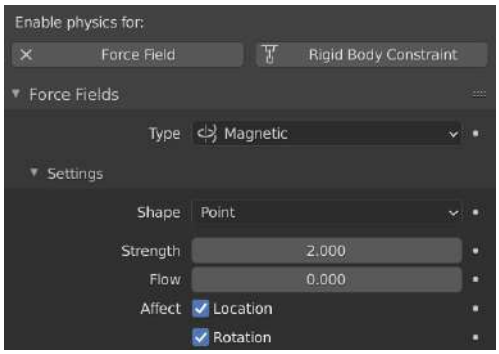


Figure 20: Magnetic physics property

Vertically above the geometric centre of the platform, a *passive rigid body* cube was created, and an empty was placed in its centre. In Bullet, a *passive* rigid body is a physics object that remains static, whereas an *active* body is dynamically simulated.

A cylinder was constructed, and a sphere was attached to its end, with an empty placed in the centre of the sphere. The 2 meshes were then *joined* to create a singular geometry, and the *origin* (centre of mass) was defined at the centre of the sphere using the empty. Thus, the mesh of the pendulum spindle and bob was created, and the spindle is effectively weightless. The pendulum was made an *active* rigid body in the *Physics Properties* tab (Figure 22b).

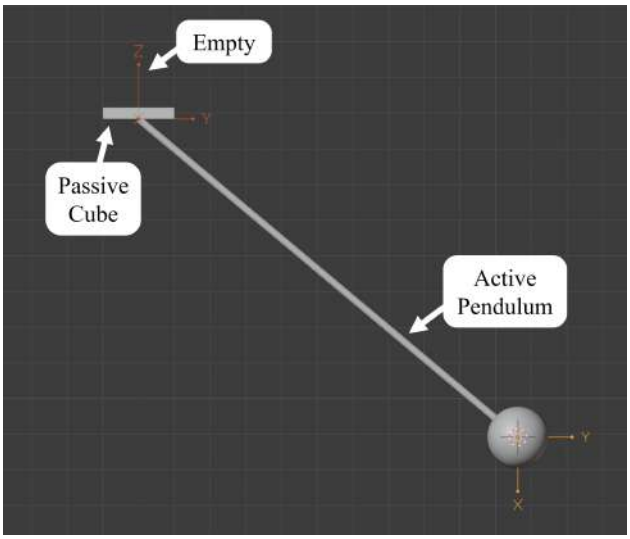
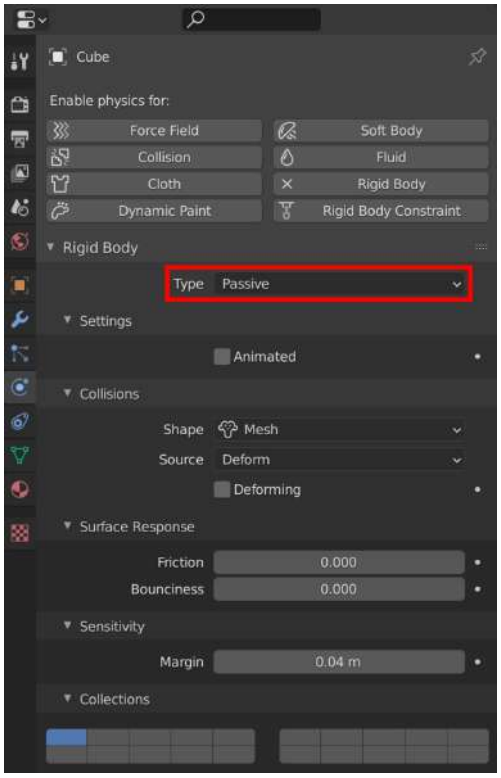
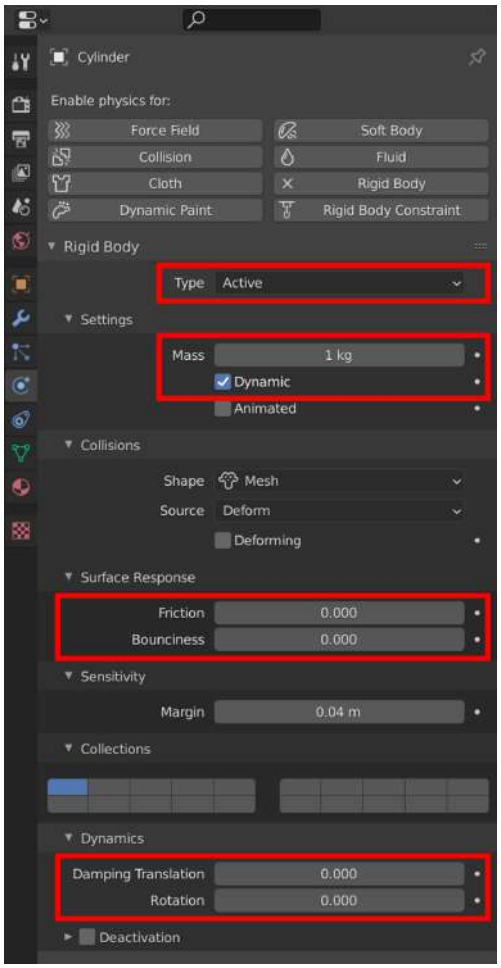


Figure 21: Passive cube, Empty and pendulum objects



(a) Passive rigid body settings



(b) Active rigid body settings, including mass, friction and damping settings

Figure 22: Physics Properties tab

To make the system dissipative or conservative, friction, damping and rotation were set in

the *Physics Properties* of the active pendulum and passive cube (Figure 22b).

The empty in the passive cube was given a *rigid body constraint* in the *Physics Properties* tab (Figure 23). The *rigid body constraint* type was set to *point*, the *first object* was set to the pendulum and the *second object* was set to the passive cube. This constraint allows the pendulum (first object) to swing with its end pivoting about the empty's origin, while disabling collisions between the pendulum and passive cube (second object).

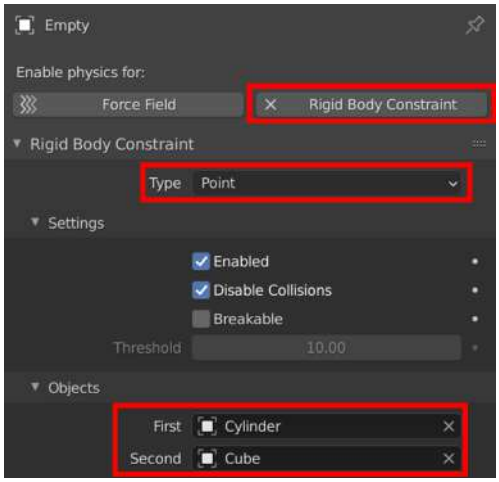
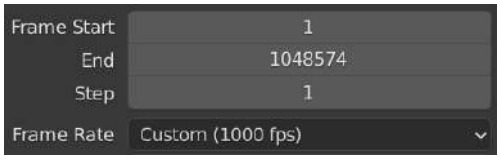
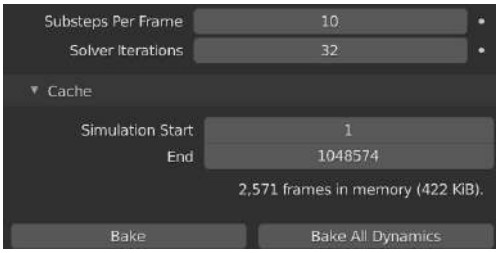


Figure 23: Rigid body constraint settings

In the *Output Properties* tab shown in Figure 24a, the last frame of the simulated animation, *frame end*, was set to 1,048,574 such that the maximum number of data points would be generated. The *framerate* was set to 1000 such that the data would be sampled 1000 times per second.



(a) Output Properties tab



(b) Rigid Body World settings in Scene Properties tab

Figure 24: Sampling and simulation parameters

In the *Scene Properties* tab shown in Figure 24b, the *substeps per frame* was set to 10 and *solver iterations* was set to 32. The *substeps per frame* is the number of times that the system is simulated per frame (each frame represents a duration of 0.001 seconds). As Bullet uses iterative methods, a higher number of solver iterations would lead to higher accuracy at the cost of computational speed.

The *Text Editor* was used to save and export the motion path. This is a text interface that

uses Python code. A script was used to save the location of the empty in the pendulum bob for each of the 1,048,574 frames.

The time between each frame is 0.001 seconds, due to the selected framerate. Therefore, the velocity and acceleration of the pendulum bob can be calculated for each frame

This process was repeated for a dissipative system with certain parameters changed. In the *Physics Properties* of the pendulum object, friction, damping and rotation were set to 0.1, 0.05, 0.05, respectively. The total number of frames was set to 200,000 as the pendulum came to rest by this point. A magnetic strength of 4 was selected for the dissipative system for the same reasons described previously.

Figure 25: 4-DOF Double Pendulum

The double pendulum is a simple yet classic example of a chaotic system. Models of this system typically limit the degrees-of-freedom (DOF) to 2 (1 for each pendulum). The model chosen to demonstrate chaos was a 4 DOF double pendulum, with 2 pendulums that are each free to move in the azimuth and elevation angular directions (Figure 26).

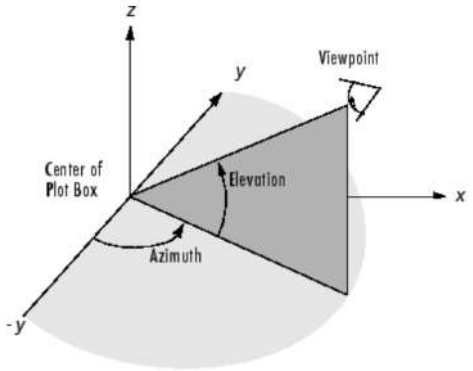


Figure 26: Azimuth and elevation angular directions [4]

The first pendulum was set up using the same method as the magnetic pendulum. The empty within the first pendulum bob was given point rigid body constraint to fix the second

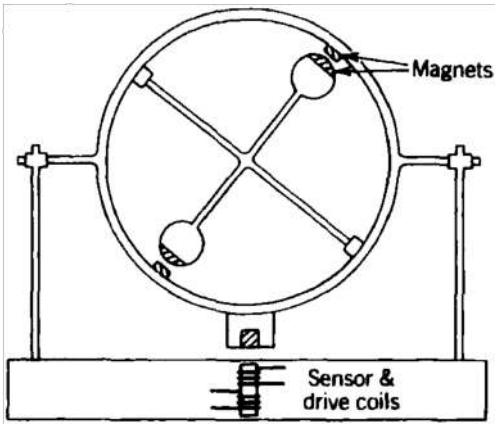
pendulum to the first and to set the point of rotation of the second pendulum bob to be the centre of the first.

Both pendulums were given a mass of 1 kg which acts at the centre of each bob, making the spindles weightless.

With 2 point constraints that allow free rotation in all directions, the system has 4 degrees of freedom.

3.2.2 Chaos in Periodically Forced Systems

2 systems were chosen to demonstrate the manifestation of chaos in periodically forced systems: the spaceballs chaotic toy and rollercoaster chaos model.



(a) Spaceballs chaotic toy [2]

(b) Spaceballs “Perpetual Motion” Toy [59]

Figure 27: Spaceballs Chaotic Toy models

The spaceballs toy involves a periodically forced outer pendulum which has an inner pendulum connected at a 45° angle to the axis of the outer pendulum. In the physical model (Figure 27a), the forcing is done by an electromagnet that is activated and repels the outer pendulum when it moves over the magnet. While Office Playground [59] calls this a ”perpetual motion toy”, it is not a perpetual motion machine as it requires batteries to power the electromagnet.

The periodic forcing was implemented in Blender by using keyframes to control the periodic motion of the outer pendulum, which was given a swinging frequency of 4.167 Hz. 2 empties were created as magnetic force fields and set as a *child* to the outer pendulum, thus fixing them to the outer pendulum at the required locations. The axis of rotation of the outer pendulum was set using an empty with a *hinge* rigid body constraint and a passive rigid body cube.

The inner pendulum was created as a combination of 3 bodies: 2 pendulum bobs and the cross-shaped spindle. Each of these had to be set to the same mass (1 kg) to prevent errors in the simulation. Each bob was given a fixed rigid body constraint to attach them to the spindle. Another empty was created and made child to the outer pendulum at the location of the inner

pendulum's axis of rotation. This empty was given a hinge rigid body constraint such that the axis of rotation of the inner pendulum is attached to the outer pendulum.

To create a periodic spaceballs model, the empties acting as magnets were deleted and the rigid body physics properties of the inner pendulum were removed. A new empty was placed in the centre of the outer pendulum with its x-axis aligned in the direction of the inner pendulum's hinge. The inner pendulum was made a child to the new empty and this empty was made a *child* to the outer pendulum. *Rotation* keyframes were applied to the x-rotation of the empty to control the inner pendulum's periodic rotation, such that it had a frequency of 2.60 Hz. The rotation was made to continue infinitely with the *Graph Editor*, which allows variables to be controlled using graphs.

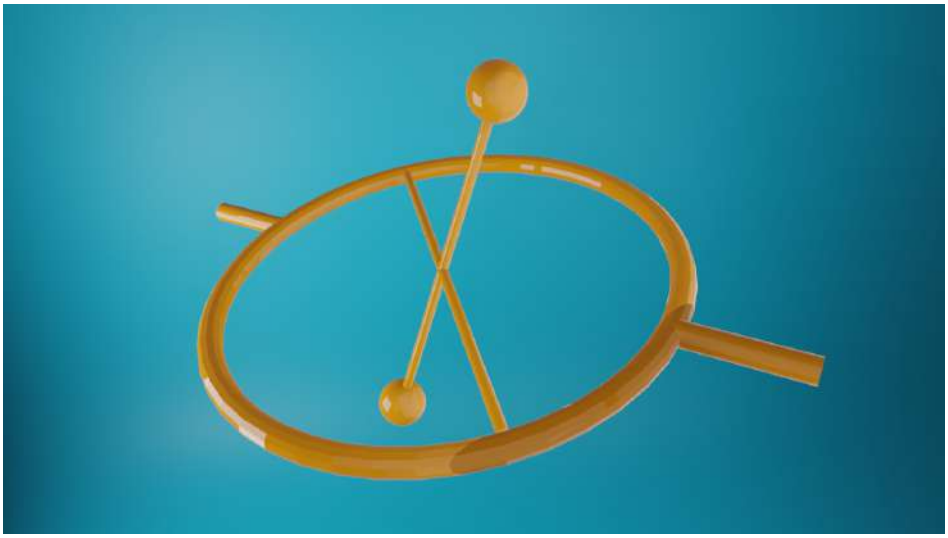


Figure 28: Spaceballs chaotic toy digital model

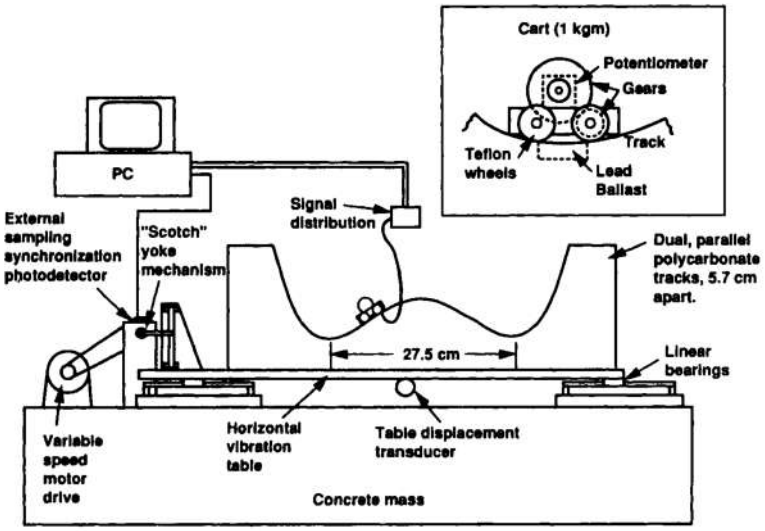


Figure 29: Rollercoaster Chaos Mechanical Apparatus [2]

The rollercoaster chaos model demonstrate the double-well potential Duffing oscillator [2]. It involves a cart which is attached but free to slide on the track with minimal friction such

that friction can be neglected. The entire track is then made to periodically oscillate such that the cart jumps chaotically between the 2 wells.

The track in Figure 29 was traced in Autodesk Inventor and exported to an STL file which was then imported into Blender and scaled to the dimensions defined in Figure 29.

The cart was modelled as a 1.8 cm diameter ball. The ball was constrained by the track by creating an upper shell which allows a 1.8 cm gap between the surface of the upper shell and surface of the track (Figure 30). Friction, damping and rotation parameters were set to 0 in Blender to eliminate all frictional effects.

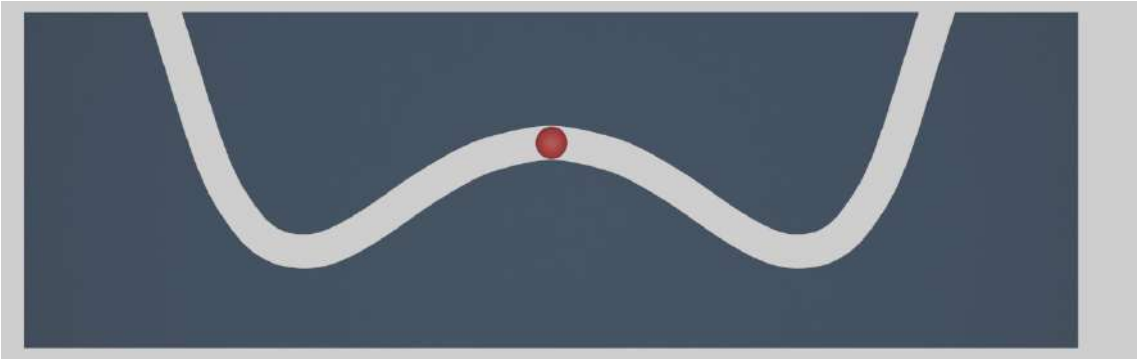


Figure 30: Rollercoaster Chaos Digital Model

3.3 Statistical Outputs from Stochastic Inputs

3.3.1 Oscillating Galton Board: Digital Experiment

Figure 31: Resulting distribution from pegs oscillating at 16 Hz

This project continues the work of Phelan [3] which investigated the effects of a biased Galton board. Phelan wrote Python code that generates the meshes and physics properties of a Galton board. With Phelan's permission and under the advice of my supervisors, this code was modified so that the pegs were made to oscillate using keyframes. This method of controlling the

pegs' motion required each peg to be defined as an *animated* rigid body and does not allow the the pegs' motion to be affected by other rigid bodies. The pegs oscillate translationally and horizontally. The frequency (f), amplitude and phase difference between rows of the oscillation were set as variables.

The phase difference refers the the phase difference between odd and even rows of pegs. The code was also modified to include a wider arrangement of bins to capture the more spread out distribution due to he oscillations. The modified script that generates the oscillating Galton board in Blender is given in the Appendix.

The framerate and number of simulated frames were set such that the keyframes could be generated correctly and the simulation would end when every ball has reached the bottom of the Galton board.

The code was placed within a loop to automate the running of simulations and saving of data. After the simulation has been completed, the horizontal locations of each ball in the last frame is saved. These coordinates are exported and saved in a CSV file as a spreadsheet. From this file, the distributions from the Blender simulations could be recreated as histograms using Matlab.

Simulations of the oscillating Galton board were conducted with a fixed amplitude. The spacing and number of pegs were also fixed and the total number of balls was 900 in each test. Each of these parameters could be altered to replicate the equipment of the mechanical experiment.

The digital approach with Blender, combined with Python scripting in its *Text Editor* allowed fast and automated digital experiments to be conducted. By contrast, mechanical experimentation requires the experimenter to be physically present to set up the experiment and manually record each result every time.

3.3.2 Oscillating Galton Board: Mechanical Experiment

The digital oscillating Galton board was recreated mechanically (Figure 32). The Galton board itself was made from 3mm acrylic Perspex cut by a laser-cutter which allowed for fast and easy fabrication. A 40 W laser was used at 5 mm/s to cut the parts with a single pass. The individual parts of the bin walls, floor and containment walls were slotted together like a puzzle and glued in place with superglue (Henkel Loctite cyanoacrylate) to create the bin assembly. A separate assembly was made for the peg array, which comprised of 2.5 mm steel dowel pegs glued to an acrylic board. The CAD drawings are provided in the Appendix.

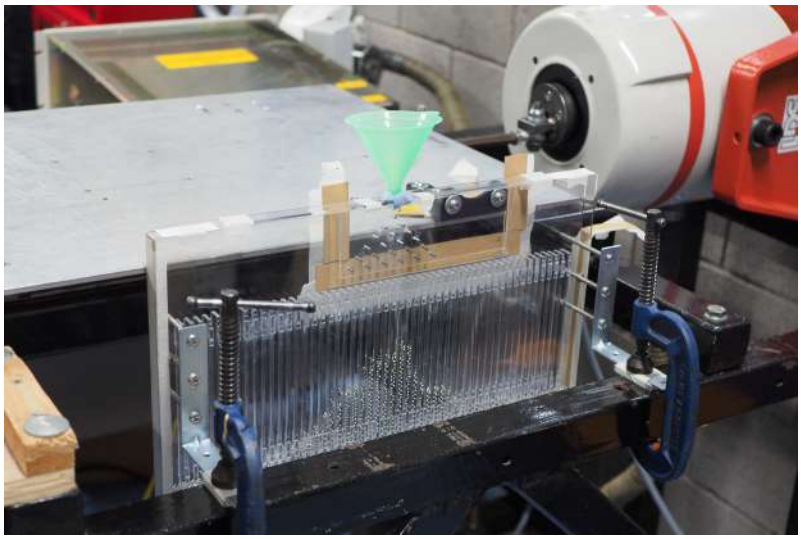


Figure 32: Mechanical experimental setup of oscillating Galton board

The peg array was made to oscillate by an LDS electrodynamic shaker which takes the inputs of frequency and an amplitude (as voltage) through a waveform generator. To constrain oscillation to a single axis, the shaker was connected to a table on rails. For a given amplitude in Volts, the shaker's oscillation amplitude in millimetres decreases as frequency increases. The maximum amplitude that could be supplied to the shaker was 5 V. Therefore, as frequency was varied, the amplitude had to be adjusted accordingly to obtain the desired oscillation displacement. Due to the constraints of the maximum Voltage amplitude, and the minimum frequency to prevent overheating, the frequencies 5-10 Hz were tested in increments of 1 Hz. The displacement was set to be approximately 3 mm, which is the diameter of the balls passed through the Galton board.

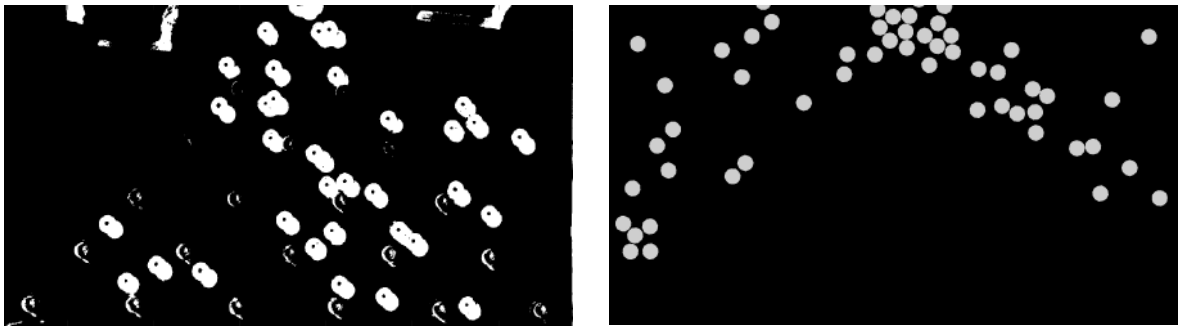
The peg array was mounted to the shaker table with a 3D-printed L-bracket and the bin assembly was mounted to the frame of the shaker with steel L-brackets. At the top of this assembly, a funnel with a 0.8 mm spout diameter was secured in place with Blu Tack directly above the centre of the equilibrium position of the peg array. 3000 3mm diameter steel balls (decanter cleaning balls) were poured through the funnel to collide with the peg array. The uniformly spaced bins collected the balls so that the resulting distributions could be seen.

To record the results of each experiment, a photograph of the final distribution was taken and slow motion video was recorded of the balls falling through the peg array. A fast shutter speed (1/4000 seconds) was used to record the video in order to "freeze" the motion of the falling balls and oscillating pegs.

The recorded video was stabilised using a video processor in Blender, which cropped each frame such that the view of the camera was stationary with respect to the peg array. These frames were then exported and analysed in Matlab to conduct a direct numerical comparison. A script in Matlab was used to remove the "mode" pixels of each frame. The "mode" pixels are those that are most common to every frame of the video, which are the pixels that make up the objects that are stationary with respect to the camera's perspective. In this way, the pegs and

background were removed from the image. A threshold filter was then applied to the frames to isolate the balls, as they appear much darker than the rest of the image. An equivalent image was generated for the digital model by applying *Index* labels to the balls in Blender.

The `entropy` function from the *Image Processing Toolbox* in Matlab was used to calculate the entropy of each frame. The entropy of an image is a measure of the extent of randomness of the image, and hence provides a direct numerical comparison between the digital and mechanical results. The image analysed was the area of the peg array (Figure 33) and the images analysed started from the first ball entering the frame and ended when the last ball exited the frame. The mechanical and digital entropies were then compared with the Kruskal-Wallis test through the `kruskalwallis` function, which takes in the 2 datasets and returns a *p-value* which is the probability that the 2 data sets came from the same distribution. The script that conducted the image processing is provided in the Appendix.



(a) Mechanical, "background" removed (b) Digital

Figure 33: Processed frames from slow-motion video of mechanical and digital experiments

The mechanical experiment assessed the accuracy of Bullet in the simulation of collisions and investigation of predictable control of stochastic inputs. This was evaluated both visually and through signal processing.

3.4 Poincaré Maps to Visually Detect Chaos

From the literature, Poincaré maps were seen to be powerful and simple tools for detecting chaos as they are able to visually rule out if a system is not chaotic with ease. On the Poincaré map, periodic, quasiperiodic and chaotic motions would likely display, discrete points, closed curves, and discontinuous points, respectively (Figure 35). When sampling a periodic system at a rate equal to the a forcing frequency, a finite number of small clusters may appear on the Poincaré map (Figure 35b).

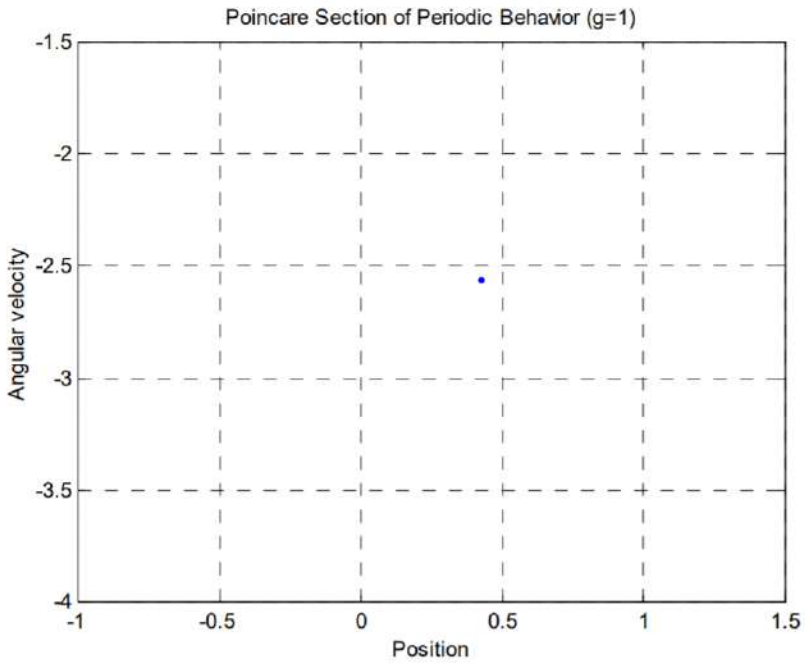
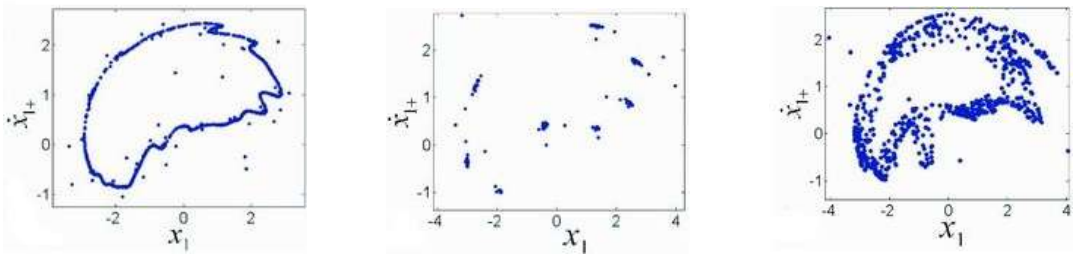


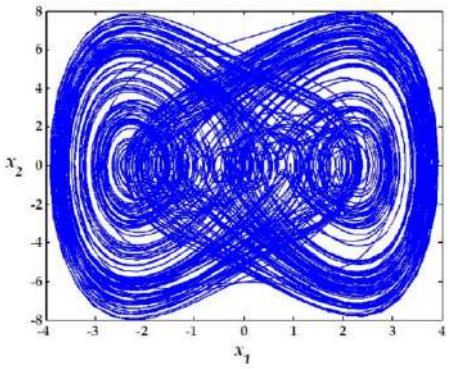
Figure 34: Poincaré map of periodic motion [5]



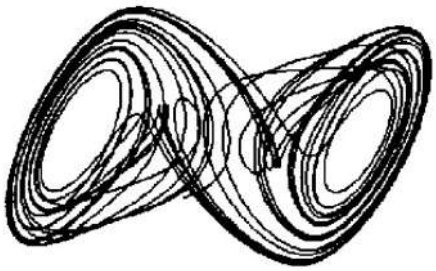
(a) Quasiperiodic (b) Periodic, Phase locked (c) Chaotic

Figure 35: Poincaré maps of chaotic and periodic TDOF vibro-impact systems [6]

To generate a Poincaré map, a phase space must first be generated from the dimensions of the system (such as cartesian or polar coordinates). Phase spaces (Figure 36) are multidimensional spaces in which each dimension is a state variable. These plots are generally created by plotting a dimension against its time derivative (e.g. $\dot{x}$ plotted against x). The state variables used in the digital experiments were the cartesian and polar positions and velocities. Cartesian and polar dimensions were chosen instead of the degrees of freedom, as this made sampling far simpler. In real systems, sampling the degrees of freedom is very challenging.



(a) Phase plane of a chaotic duffing oscillator [1]



(b) Phase plane of a chaotic double scroll system [60]

Figure 36: Phase planes of chaotic systems from literature.

Matlab, a software that most undergraduate engineers have experience with, was used to analyse the exported motion paths and calculate the phase spaces of the simulated systems. The motion path contains the cartesian coordinates of an object of interest in each simulated frame. Hence, if the framerate was set to 1000 FPS, then there is a duration of 0.001 seconds between each frame. Therefore, the velocity, acceleration and jerk can be calculated for each frame by calculating the difference between the current value and the value of the subsequent frame, and dividing by 0.001 s. For example, velocity at a given point, $\dot{x}_n$, can be calculated with the current displacement, x_n, and subsequent displacement, x_{n+1}, with equation 3:

$$\dot{x}_n = \frac{x_{n+1} - x_n}{0.001} \tag{3}$$

When a phase space has been generated, a Poincaré map can be made using a cutting plane. In this method, the phase space is sampled when its trajectory intersects the plane. A sample is only taken when the phase space trajectory enters 1 side of the cutting plane, but not the other side, as shown by Figure 37

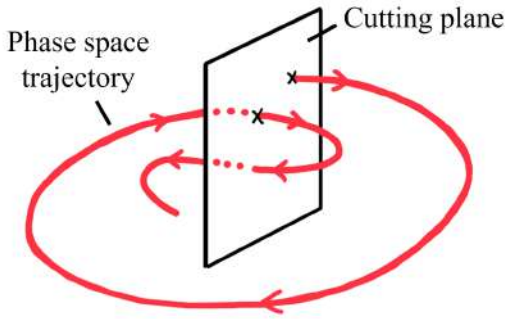


Figure 37: Cutting plane method of generating Poincare map

To create the Poincaré map in Matlab, a cutting plane was defined by its a normal vector and a point on the plane. A loop is then used to go through each straight line segment between

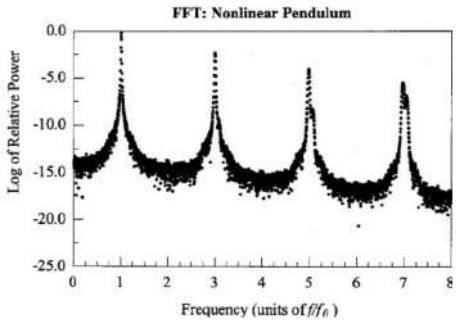
2 points of the phase space trajectory, and to determine whether the line segment intersects the cutting plane.

As the intersection is only plotted for one direction of the trajectory (i.e. entering one side of the plane and not the other, as shown by Figure 37), it is required to determine the direction of the trajectory that intersects the plane. To do this, the angle between the normal vector and the trajectory line segment is computed. If the angle is acute, then the trajectory is travelling in the direction of interest.

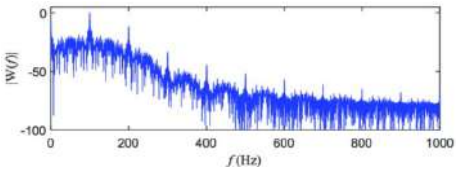
Each point of intersection is stored and plotted to form the Poincaré map. A marker of chaos is detected if the Poincaré map does not exhibit repeating patterns, a singular point, or discrete clusters. The script that computes the Poincaré map using the cutting-plane method is provided in the Appendix.

A Poincaré map can also be generated by sampling the phase space at specific sampling intervals. For periodic or quasiperiodic systems, this can be done with a sampling rate equal to a forcing frequency in the system.

Another graphical method to identify chaos from the calculated velocities is to plot the power spectrum from the Fourier transform. Matlab was able to do this using its built-in fast Fourier transform (FFT) function. Periodic motion can be visually distinguished from chaotic motion as periodic Fourier spectra show discrete peaks whereas chaotic Fourier spectra are continuous.



(a) Fourier spectrum of periodic nonlinear motion [61]



(b) Fourier spectrum of chaotic motion [62]

Figure 38: Fourier spectra of nonlinear systems from literature

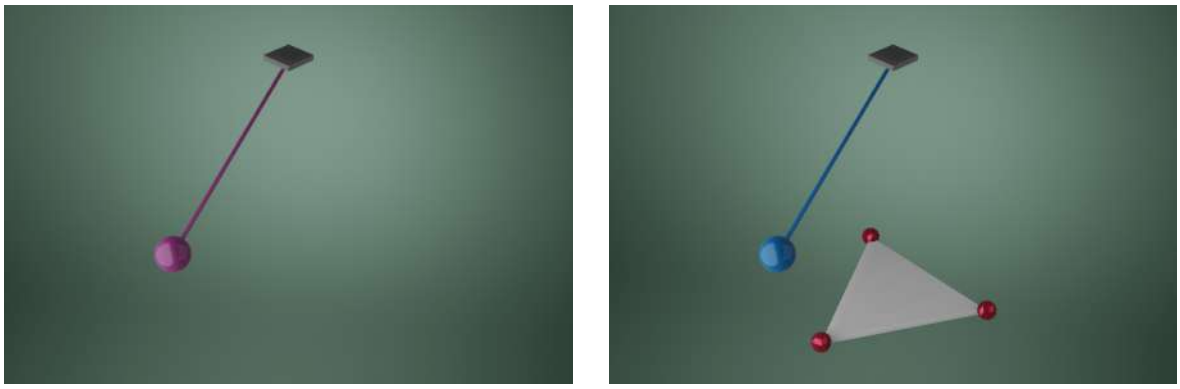
These methods of plotting the Poincaré map allow the motion path of any system to be quickly analysed. The visual and graphical nature of the results make this method well suited for a laboratory to demonstrate nonlinear dynamics and chaos to students.

4 Results and Discussion

4.1 Phase Planes and Fourier Spectra of Nonlinear Systems and Periodic Systems

Phase planes were generated for the simulated systems. These plots are useful to distinguish between linear, nonlinear, chaotic and non-chaotic systems as periodic motion would appear as a closed curve, while chaotic motion displays no repeating orbits and would fill a section of the phase space.

As a benchmark for the analysis, a simple pendulum was simulated to compare the results of a periodic system to a nonlinear chaotic system. The simple pendulum was made such that it had exactly the same mass, length, shape and starting position as the magnetic pendulum, so that it was the same as the magnetic pendulum system except that it excluded any magnets (Figure 39). The periodic motion was naturally generated through simulation with the Bullet rigid body physics engine, with no keyframes or pre-defined motion path.



(a) Simple Pendulum (b) Magnetic Pendulum

Figure 39: 2-DOF Single Pendulums

The simple pendulum's phase spaces showed closed curves that do not cross themselves (Figure 40) whereas the nonlinear magnetic pendulum had phase spaces that do cross themselves and appeared to fill a section of the phase space (Figure 41). The same characteristics were seen in the phase plane of the dissipative magnetic pendulum (Figure 42), in which friction and damping bring the pendulum to rest after some time. However, the filling in of area was not observed in the dissipative magnetic pendulum's phase plane.

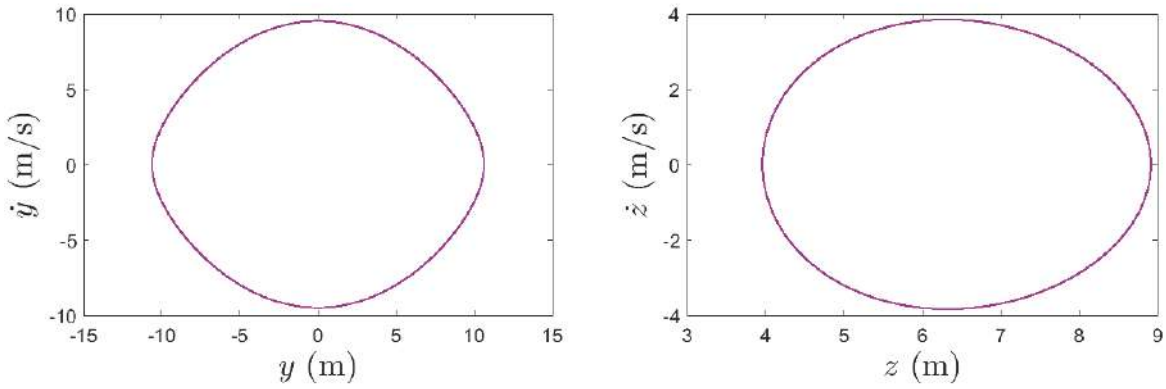


Figure 40: Simple Pendulum Phase Planes

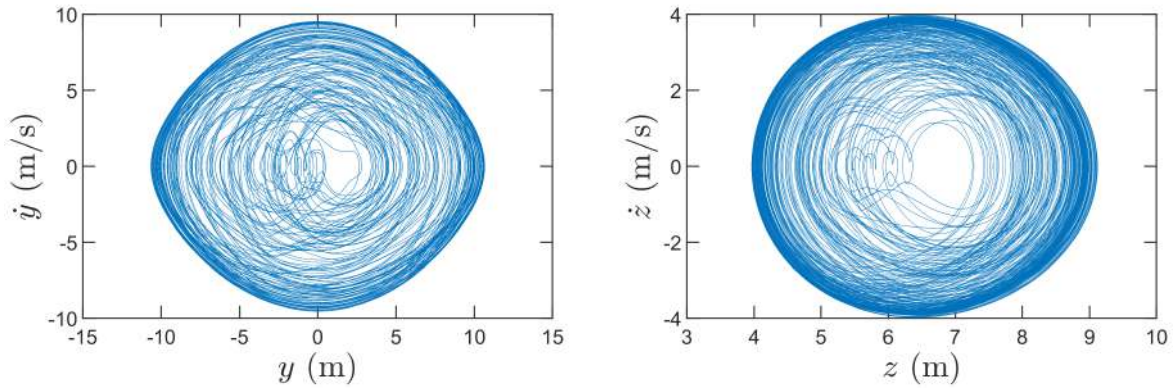


Figure 41: Conservative Magnetic Pendulum Phase Planes

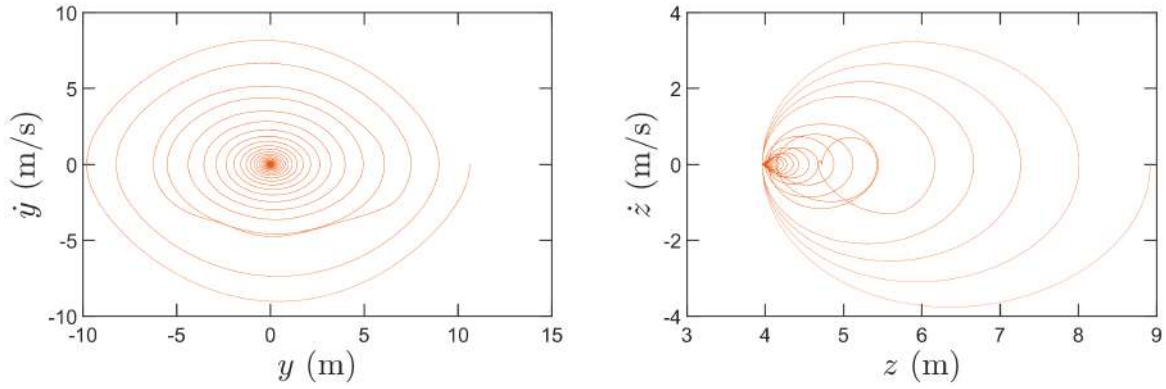
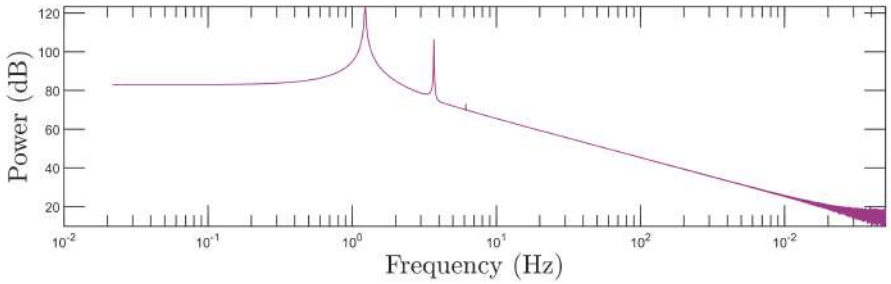


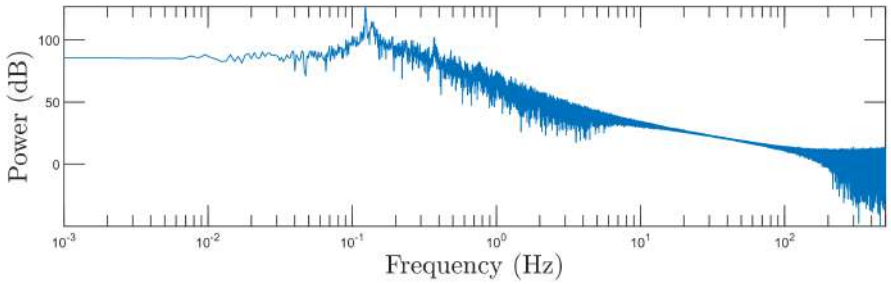
Figure 42: Dissipative Magnetic Pendulum Phase Planes

Further differences between periodic and chaotic motions can be identified in the Fourier spectra, as chaotic power spectra are continuous while periodic spectra show discrete spikes. This marker of chaos from theory was able to be demonstrated by the simple and magnetic

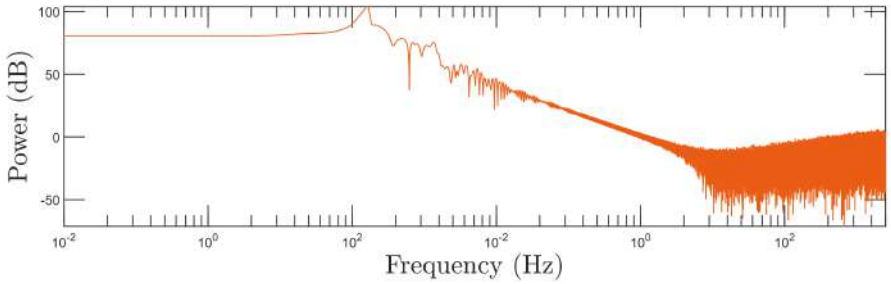
pendulums (Figure 43). In Figure 43a, the Fourier spectrum peaked at the frequency of the simple pendulum's swing. The magnetic pendulums' frequency spectra display broad band continuous frequency, which was expected from the literature review.



(a) Simple Pendulum



(b) Conservative Magnetic Pendulum



(c) Dissipative Magnetic Pendulum

Figure 43: Fourier Spectra of Periodic and Chaotic Pendulums

An animation of the magnetic pendulum can be found at https://youtu.be/qCQgsLkKxQU.

The phase planes of the double pendulum also showed trajectories that cross themselves and fill an area of the phase plane. An animation of the 4-DOF double pendulum can be found at https://youtu.be/cMQF8LBOw8M.

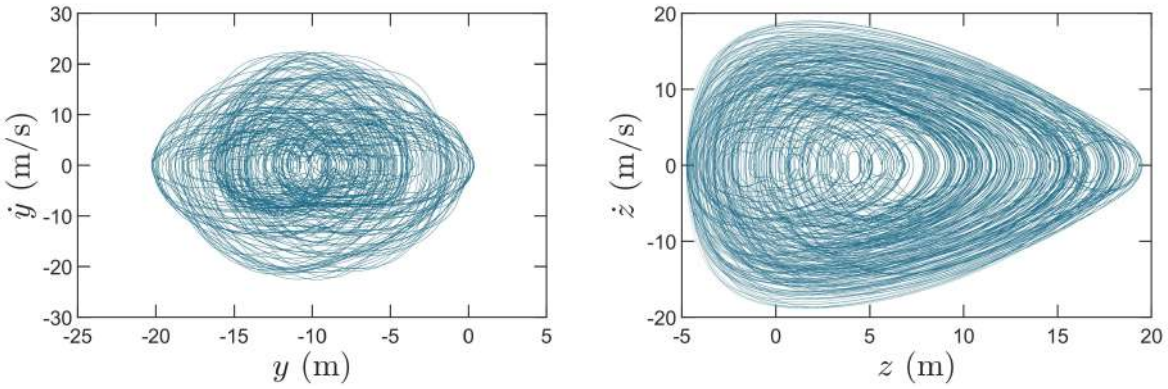


Figure 44: Double Pendulum Phase Planes

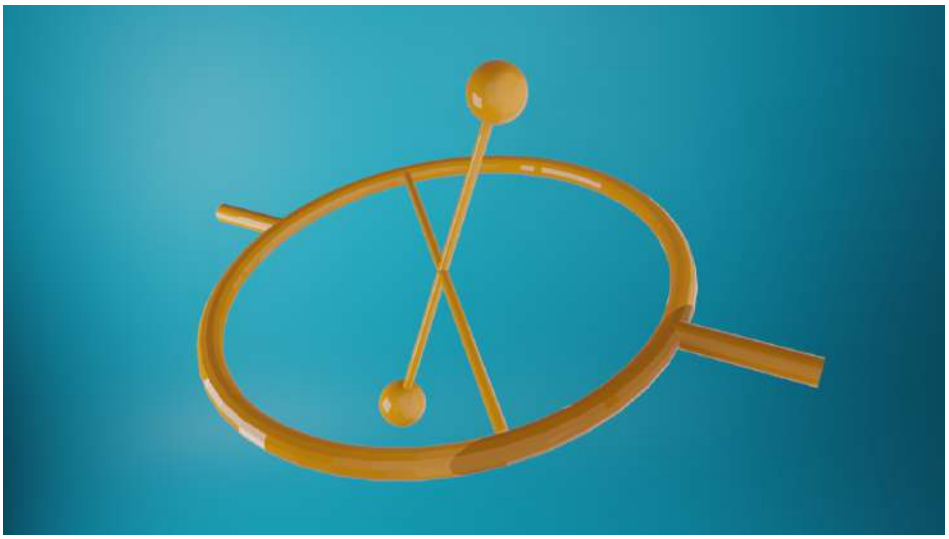


Figure 45: Spaceballs model

For the spaceballs model, the phase space was generated for one of the bobs of the inner pendulum. A chaotic model was made for which magnets and periodic forcing introduce nonlinearities. A periodic model was made in which the inner pendulum was rotated at a constant frequency of 2.6 Hz and was not affected by magnets. In this model, the hinge constraint in Blender was found to be imperfect and play was found in the hinge of the inner pendulum, which introduced an additional source of nonlinearity (Figure 46).

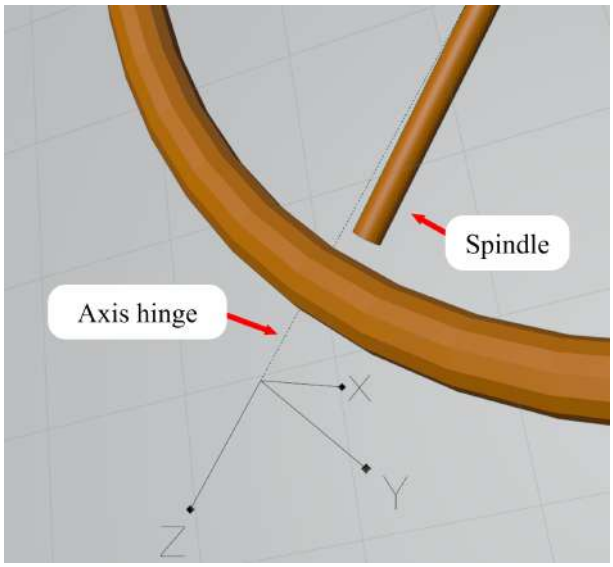


Figure 46: Screenshot of hinge in spaceballs model showing spindle detached from its hinge

The phase plane of the chaotic spaceballs showed a significant area of the phase space being filled up, whereas the periodic model has phase planes that showed curves that are significantly more ordered. Both of these phase plane trajectories cross themselves, which theory suggested would occur in nonlinear systems. At a first glance, one might mistake the phase space of the periodic spaceballs to be chaotic. For this reason, Poincaré maps are required to more thoroughly inspect for chaos.

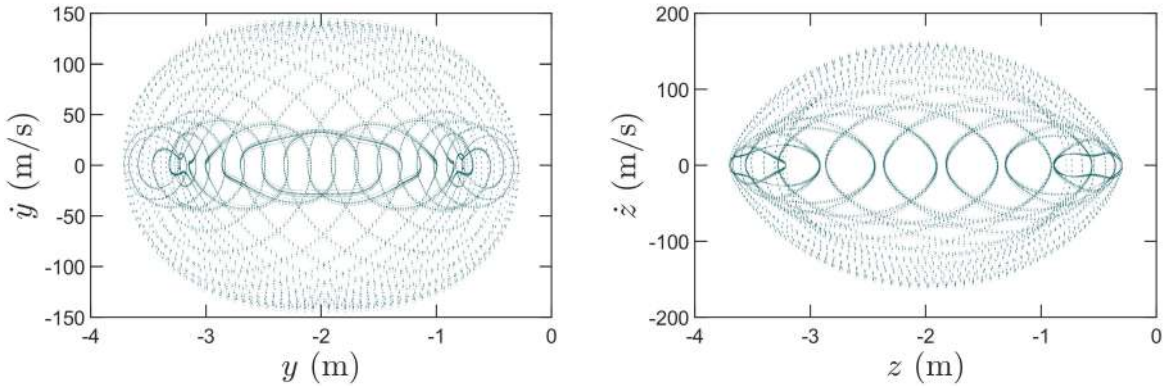


Figure 47: Periodic Spaceballs Phase Planes

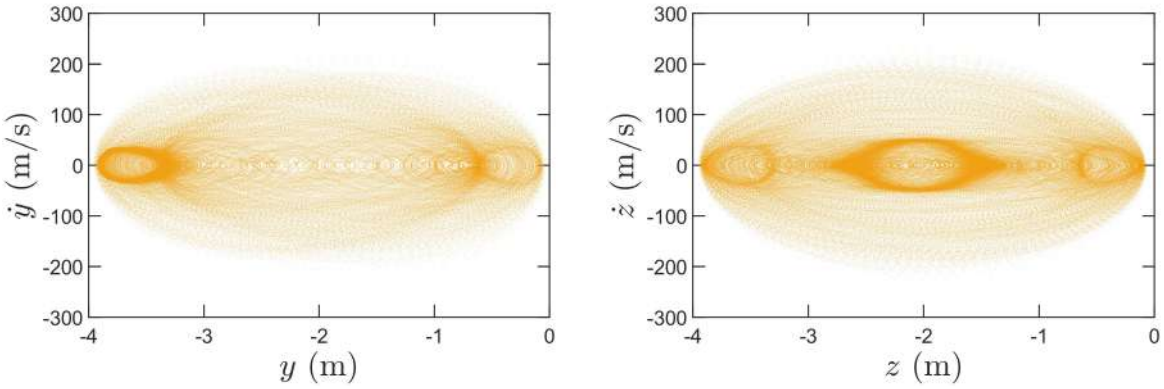
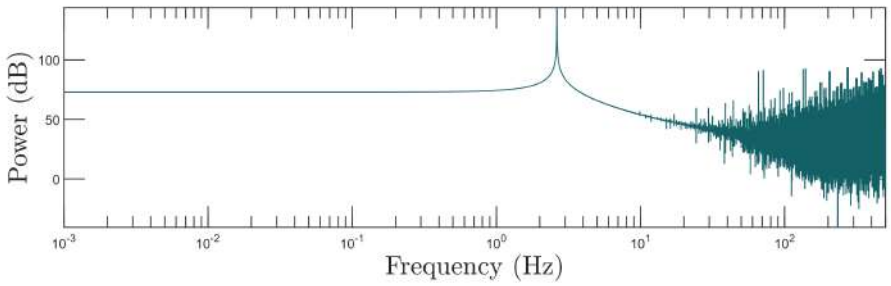
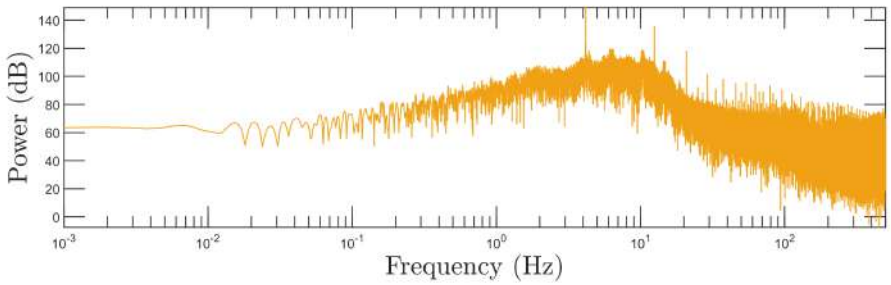


Figure 48: Chaotic Spaceballs Phase Planes

The Fourier spectrum of the periodic spaceballs model showed a discrete peak at 2.63 Hz, the frequency at which the inner pendulum was made to rotate. Like the other models, and consistent with theory, the chaotic model showed a continuous Fourier spectrum (Figure 49).



(a) Periodic spaceballs. Discrete peak at 2.63 Hz



(b) Chaotic spaceballs

Figure 49: Fourier Spectra of Periodic and Chaotic Spaceballs Models

In the rollercoaster chaos model described by Moon [2], a ball is constrained to a track with 2 troughs (Figure 50). When the trough is made to oscillate, the ball jumps chaotically between the 2 troughs.

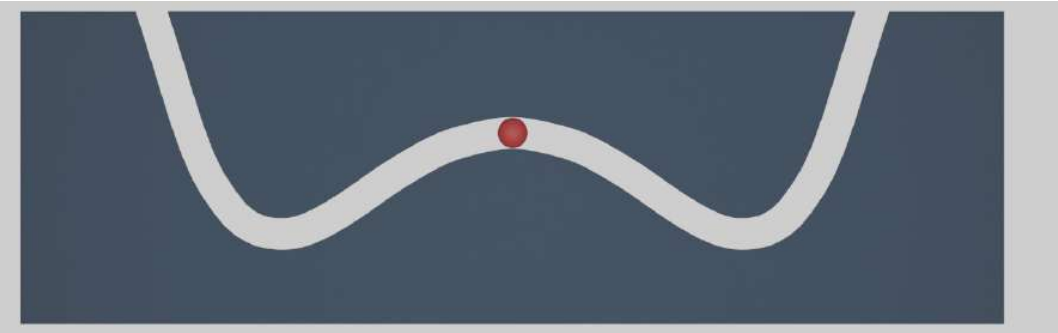


Figure 50: Rollercoaster chaos model

The rollercoaster chaos model could not be successfully created in Blender with adequate accuracy. In the Blender model, the ball's starting position was on top of the crest between the 2 troughs. The ball jumped between the 2 troughs for only certain values and combinations of oscillation frequency and amplitude. When the ball was made to jump unpredictably between the 2 troughs, it appeared to roll along the track. However, as there was no friction or damping introduced to the system, there should have been no rolling of the ball.

Furthermore, the unpredictable jumping only occurred for a short duration before the ball stopped jumping and settled in one of the troughs for the remainder of the simulation. This behaviour was shown in the time history of the ball's velocity(Figure 51), where there was intermittent chaos (bursts of chaos between periodic motion), which eventually settled to 0-velocity relative to the oscillating track.

The apparent dissipation observed in the rolling was likely the cause of this. The source of the dissipation could have been the finite resolution of the track's mesh. The constraint of the ball tightly against the track could have caused the mesh of the ball to 'clip' through the walls of the track, which would lead to dissipation. The rolling can be viewed at https://youtu.be/ON72zaZFIys. A video of the rollercoaster chaos simulation can be viewed at https://youtu.be/Qw2Oy8_TA64.

The Fourier spectrum (Figure 52) showed that the system is not periodic as the frequency spectrum is continuous. If the system were periodic, one might have expected the Fourier spectrum to show a discrete peak at the forcing frequency of the track. The phase planes showed results consistent with the other chaotic nonlinear systems (Figure 53).

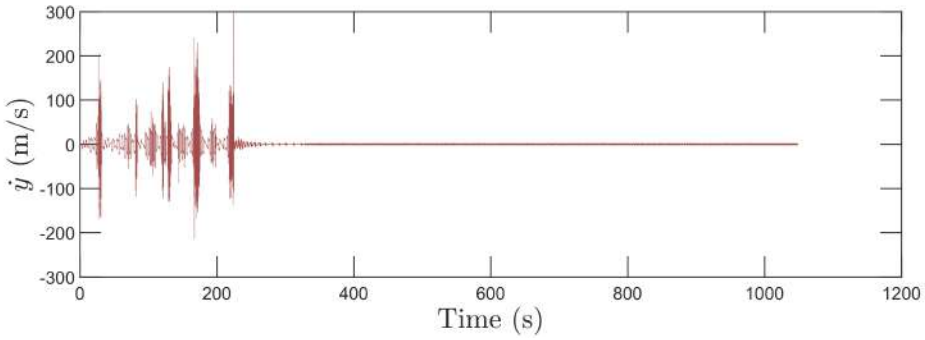


Figure 51: Time history of the ball's velocity in the rollercoaster chaos model

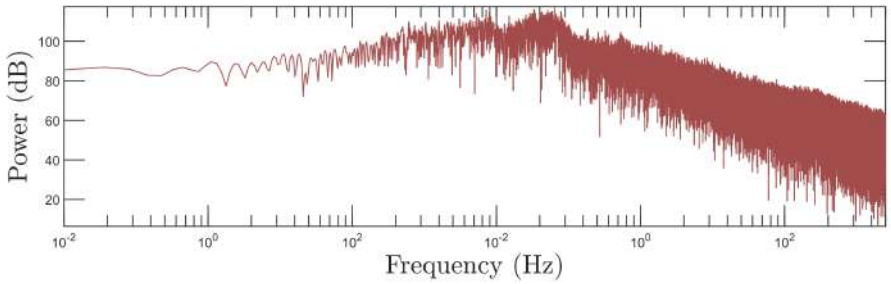


Figure 52: Fourier Spectrum of the ball's velocity in the rollercoaster chaos model

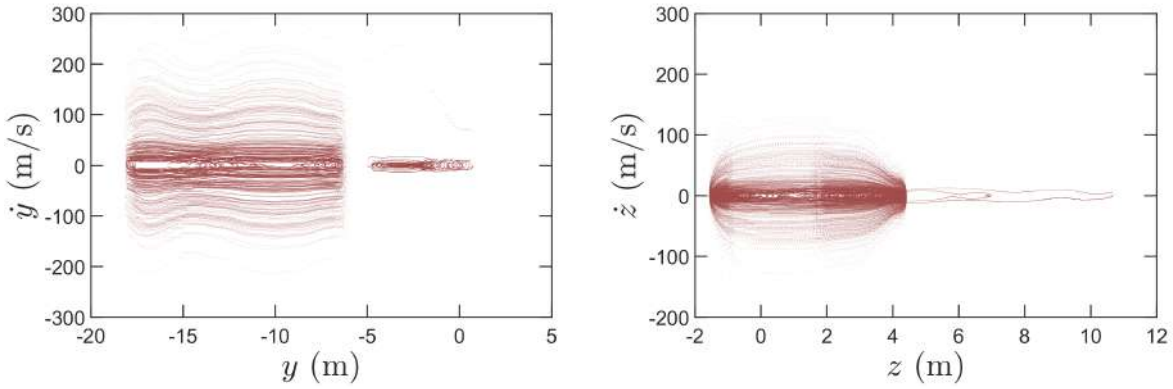


Figure 53: Rollercoaster chaos model phase planes

All of these results (except those of the simple pendulum) showed phase plane trajectories that crossed themselves, which is a characteristic that is only possible in nonlinear systems. All of the chaotic simulations showed an area of the phase plane getting filled up, except the dissipative magnetic pendulum, which was the only dissipative system investigated. This was likely due to the relatively high dissipation that limited the possible number of separate points on the phase plane, which would have otherwise filled up more area on the plane.

From these results, it was clear that Blender was able to recreate systems that were not only realistic in appearance, but that were also dynamically and theoretically accurate when analysed by practical methods used in engineering real systems. This was especially evident

when the chaotic systems were compared to the periodic versions, as demonstrated by the simple/magnetic pendulum and spaceballs models.

The phase planes of the chaotic systems show closed curves which Moon [2] suggests would mean that the simulated systems are not truly chaotic, as there is some amount of repeatability. However, experimental results from Zaher [1] showed a chaotic phase plane with a closed curve (Figure 9), which implies that the simulated results may be accurate indeed.

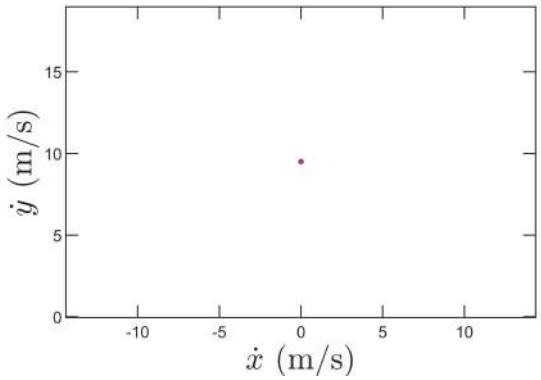
Limitations to Bullet and Blender's simulation were discovered. The rollercoaster chaos model demonstrated Bullet's inability to accurately simulate frictionless dynamic contact between 2 rigid bodies, as there was evident dissipation and rolling in a system that was supposedly frictionless. In addition to this, the hinge constraint in Blender was found to be imperfect, as it had play and hence introduced unintended nonlinearity to the system (Figure 46).

4.2 Poincaré Maps of Nonlinear Systems and Periodic Systems

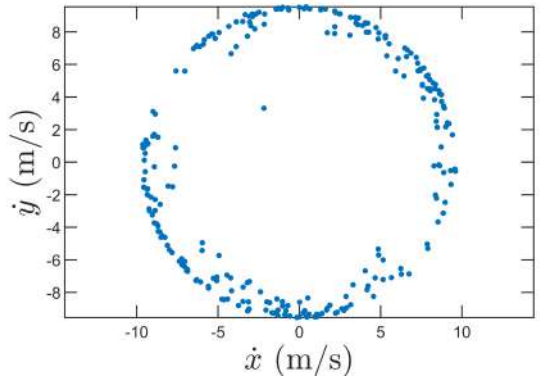
The literature suggests that while phase planes can display useful markers of chaos, a Poincaré map is required to more thoroughly detect chaos [2]. The Poincaré map of a periodic system will form single dots on the chosen section. On the Poincaré map, a conservative chaotic system would show unorganised points and dissipative chaotic motions would show highly organised points and can display fractal structures which are strongly indicative of chaos.

The trajectories of the phase planes were cut by a plane with normal in the z-direction. The trajectory was sampled where it enters the bottom of the plane.

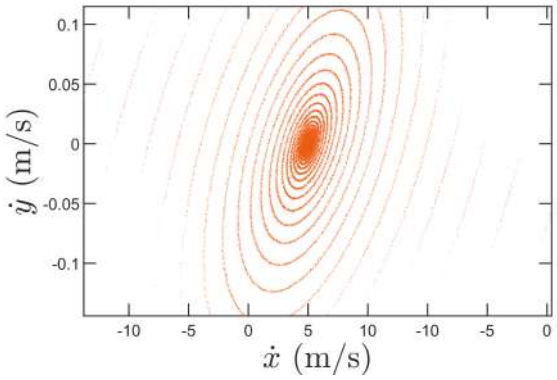
The Poincaré map of the conservative magnetic pendulum showed relatively unorganised points whereas the periodic pendulum showed 2 singular points on its Poincaré map. While the conservative map showed discontinuous and unorganised points, the dissipative magnetic pendulum showed organised points with a spiral fractal at the centre of the map. A spiral is considered a fractal structure as its appearance is the same at any scale. However, any dissipative system would also display an elliptical spiral that converges towards 0-velocity in its phase space and a spiral would be plotted onto the Poincaré map. Each of these results replicate the suggestions from literature [2, 5].



(a) Simple pendulum



(b) Conservative magnetic pendulum



(c) Dissipative magnetic pendulum

Figure 54: Poincaré maps of 2-DOF pendulums

The conservative chaotic system of the double pendulum also had a Poincaré map that showed discontinuous points.

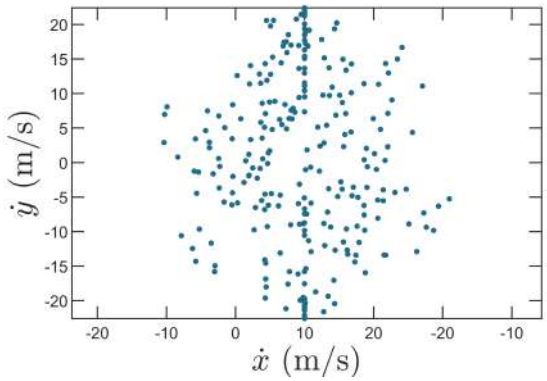
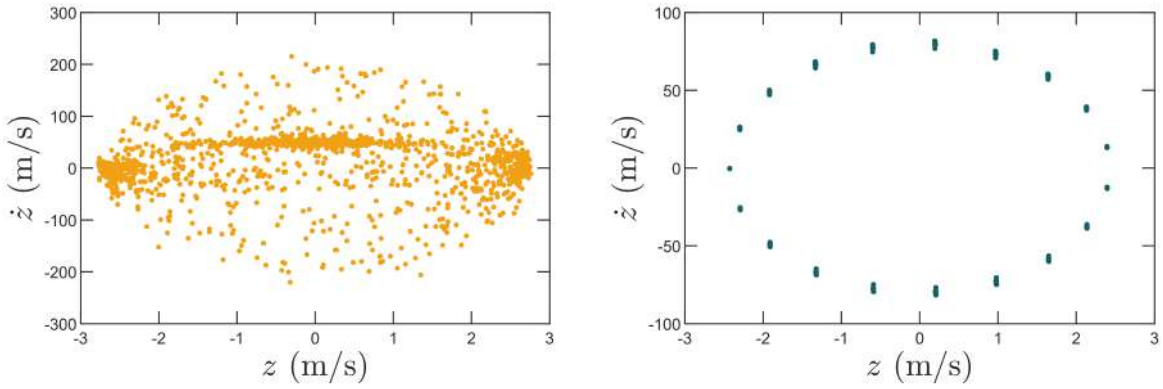


Figure 55: Poincaré map of double pendulum

As the periodic spaceballs model is periodically forced, it was appropriate to sample the phase space at a rate equal to the forcing frequency to create the Poincaré map. This sampling was done for the z-dimension (Figure 56). The chaotic and periodic spaceballs models were each forced with a frequency of 4.167 Hz, with a period of 240 frames. Hence, every 240th

value of z and $\dot{z}$ of the phase plane was plotted to create the Poincaré map.



(a) Chaotic spaceballs

(b) Periodic spaceballs

Figure 56: Poincaré maps of spaceballs models sampled at forcing frequency

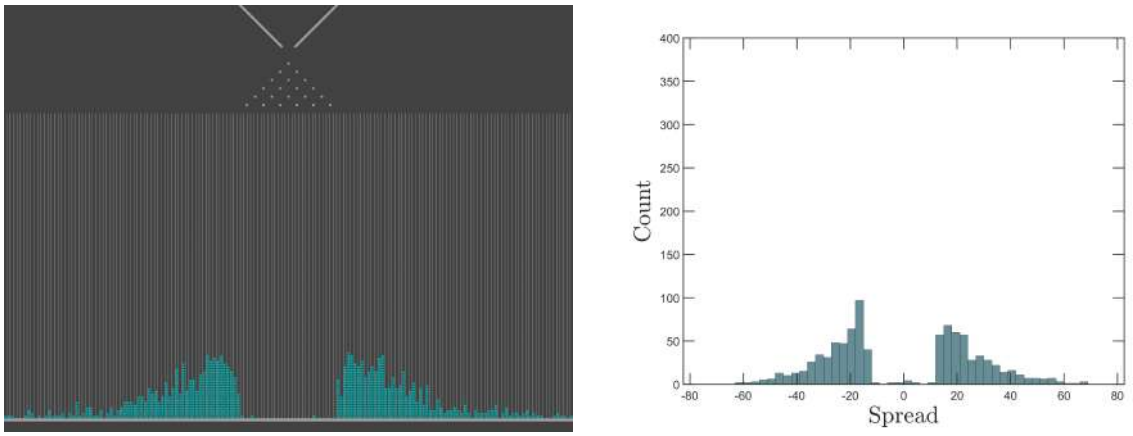
The Poincaré maps of the spaceballs model showed the characteristics suggested by Cheng et al. [6], as shown by Figure 56 and Figure 35. The Poincaré map showed several organised clusters of points, whereas the conservative chaotic system showed a "cloud of unorganised points", which was expected due to the assertion by Moon [2].

With the use of Poincaré maps, Bullet was successful in identifying markers of chaos in nonlinear systems that were forced both autonomously and periodically. The results of these maps replicated what was suggested by theory and experimental results from literature. Therefore, the digital laboratory environment performed the same function as a traditional laboratory, in that it can make a student create a system that demonstrates how theory is reproduced in "reality". Like the results of the phase planes and Fourier spectra, Poincaré maps were effective in distinguishing between periodic and chaotic systems. The visual nature of the graphical analysis would make these experiments an effective learning experience for students in undergraduate engineering.

While these markers of chaos were found, additional more thorough methods, such as Lyapunov exponents, should be used to determine if these systems are chaotic with greater certainty. However, Moon [2] suggested using at least 2 methods to identify chaos, and 3 methods were implemented in these experiments: phase planes, Fourier spectra and Poincaré maps.

4.3 Output Control from Stochastic Input

To assess the utility of Blender and Bullet in dynamical experimentation, they were used to investigate predictable control over stochastic input in the oscillating Galton board (OGB). To analyse the results from these simulations, the final locations of each ball was exported to a csv file and plotted as a histogram in Matlab, as shown by Figure 57.



(a) Final distribution in Blender

(b) Plotted histogram in Matlab

Figure 57: Histograms in Matlab to analyse the distributions formed in simulations

4.3.1 Effect of Frequency in the Oscillating Galton Board

The script was run such that automated simulations of the OGB were carried out with frequency varied and all other variables kept constant to investigate the effects of the peg oscillation frequency on the resulting distribution. Simulations were conducted from 0 Hz to 2048 Hz, which was the highest oscillation frequency simulated due to hardware limitation. From 0-256 Hz, the tests were conducted in increments of 1 Hz. From 256-2047 Hz, the frequency tested was increased by a factor of 2 with each test such that 256, 512, 1024 and 2048 Hz were tested. In these tests, framerate (FPS) was set to 4 times the oscillation frequency.

In Blender version 2.90, apparent internal resonance was discovered in the form of narrowing.

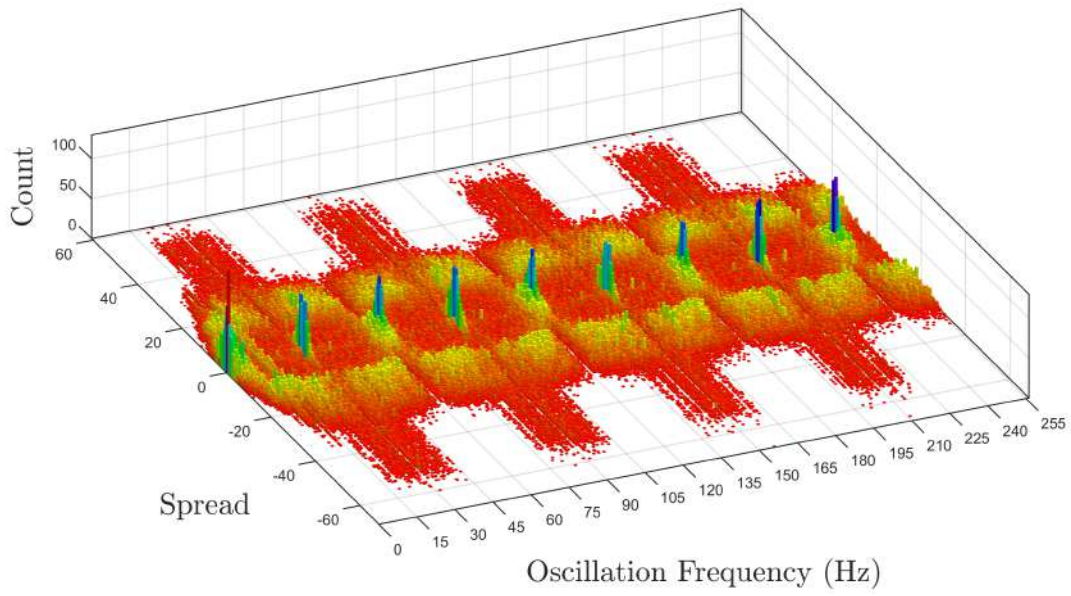


Figure 58: Stacked distribution histograms, 0-256 Hz, 60 SPS.

At 0 Hz, a roughly normal distribution forms with a peak at the centre, as expected from the standard static Galton board.

As the frequency of oscillation was increased from 0 Hz, the distribution was observed to split and form 2 peaks on either side of the centre. As the frequency was increased to 29 Hz, the spread of the distribution increased. However, at 30 Hz, the distribution instantaneously narrowed and formed a similar normal distribution to the result of 0 Hz.

Between 30 Hz and 31 Hz, the distribution instantaneously spread to the same extent as 29 Hz. As the frequency increased from 31 to 59 Hz, the distribution gradually narrowed at the same rate as it spread out in the 1-29 Hz region. At 60 Hz, the distribution instantaneously narrowed to the normal distribution once again.

This narrowing and spreading cycle repeated every 60 Hz. The transition into the narrowing was more gradual in frequency multiples of 60 Hz, while the transition into the narrowing was highly abrupt in frequency multiples of 30 Hz (when the frequency is not also a multiple of 60 Hz).

The cycle of narrowing and spreading as frequency from 0 to 240 Hz was made into an animation, which can be found at https://youtu.be/gZaxhf_gXlA.

The first series of experiments was conducted at 60 steps per second (SPS). The distribution was observed to be highly dependent on the SPS setting in the *Scene Properties* of Blender. The steps per second is the number of calculations made in one second of the simulation. Increasing this number makes the simulation more accurate. The frequencies at which the narrowing occurred appeared to be highly dependent on the selected SPS. Narrowing was found to occur when the oscillation frequency was an integer multiple of the SPS or an integer multiple of 0.5 of the SPS. The convenience of this relationship suggested to the author that the internal resonance was a numerical artefact. However, the project supervisor maintained that internal resonance was a physical phenomenon occurring within the system.

Therefore, to investigate the effect of SPS on the internal resonance, the experiments were repeated at 131 SPS. A prime number was selected to limit the number of factors and multiples of the SPS in the range of frequencies tested. In these tests of 0-256 Hz, resonance occurred only at 131 SPS, with narrowing gradually occurring near the multiples of 65.5 Hz (half of 131 SPS). Moreover, resonance was found to occur at 400 Hz at 100 SPS, but not at 60 SPS. Finally, an animation was created for the simulation of 30 Hz at 60 SPS, which showed balls passing through the straight through pegs without colliding.

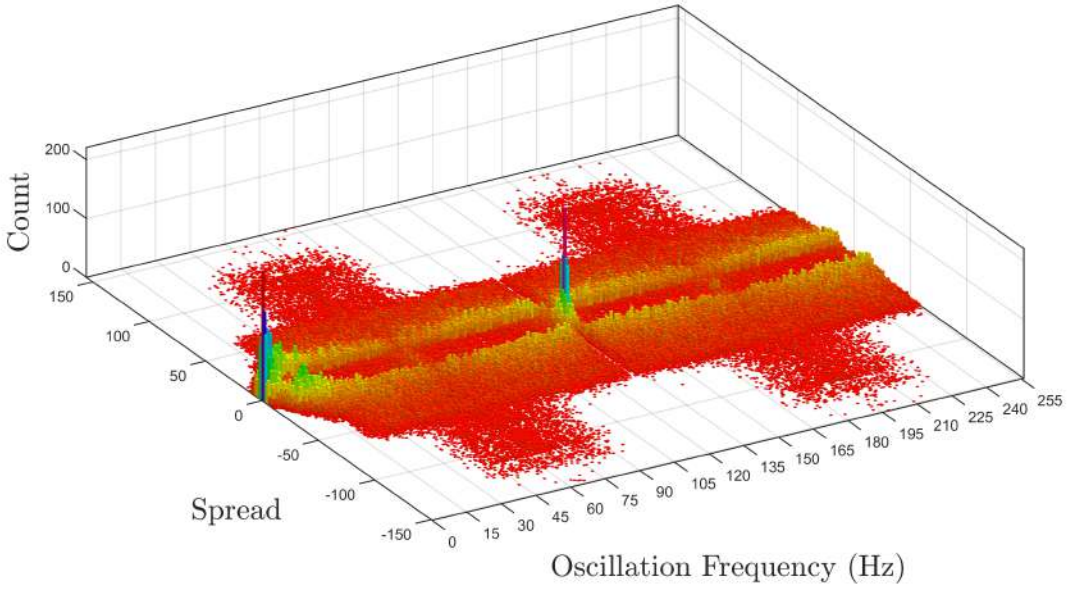
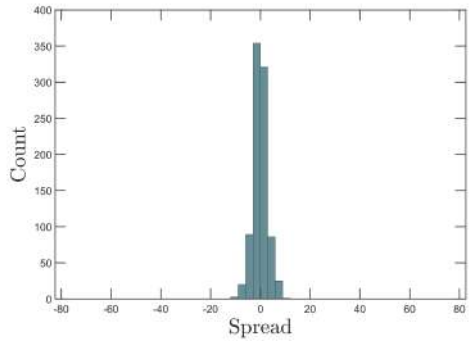
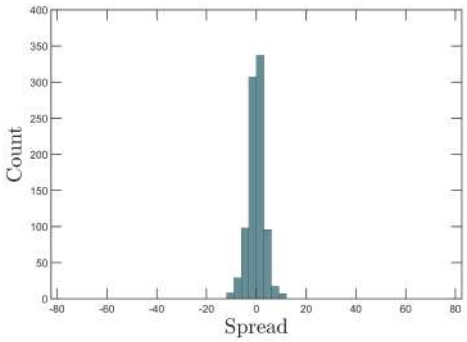


Figure 59: Stacked distribution histograms, 0-256 Hz, 131 SPS

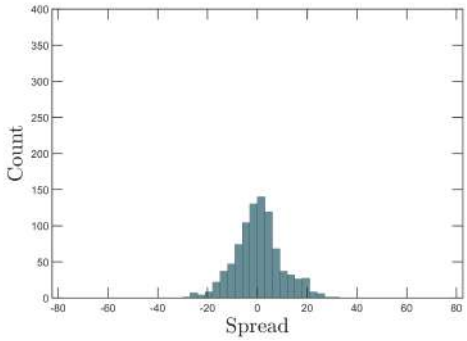
To investigate the effects of framerate (frames per second, FPS), 3 simulations were conducted at the same frequency and SPS of 60 Hz and 60 SPS. The first test was conducted at 240 FPS, which was 4 times the oscillation frequency, as was done on all the previous tests. The second test was conducted at 350 FPS (5.833 times the oscillation frequency) and the third test was conducted at 360 FPS (6 times the frequency). Resonance did not occur at 60 Hz when the framerate was 350 FPS, as shown by Figure 60c. However, resonance occurred in the 2 tests where the framerate was an integer multiple of the oscillation frequency.



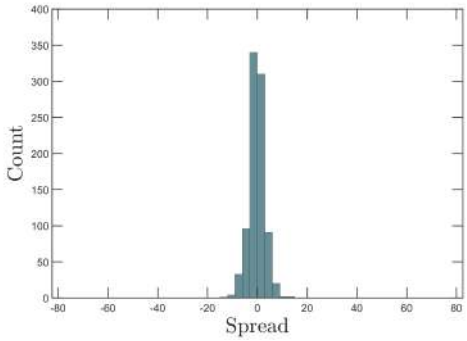
(a) Control
(0 Hz, 60 SPS, 24 FPS)



(b) 240 FPS (framerate = $4f$)



(c) 350 FPS (framerate = $5.833f$)

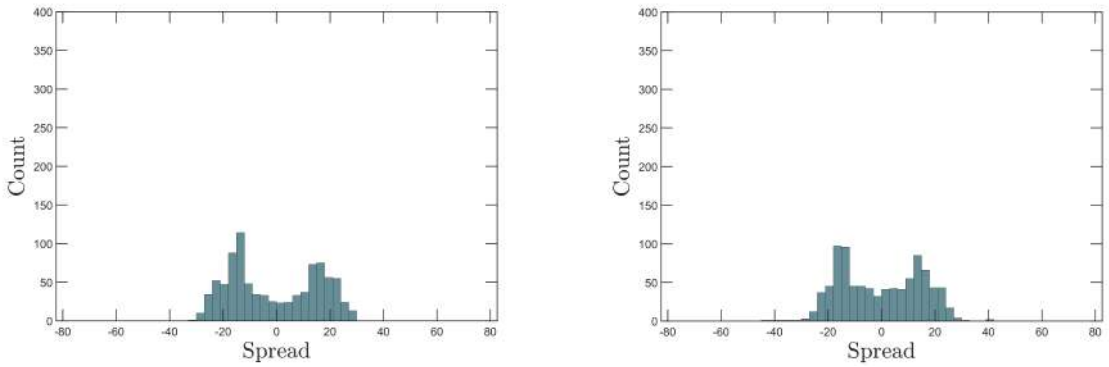


(d) 360 FPS (framerate = $6f$)

Figure 60: Effects of framerate (FPS) on distribution at 60 Hz, 60 FPS

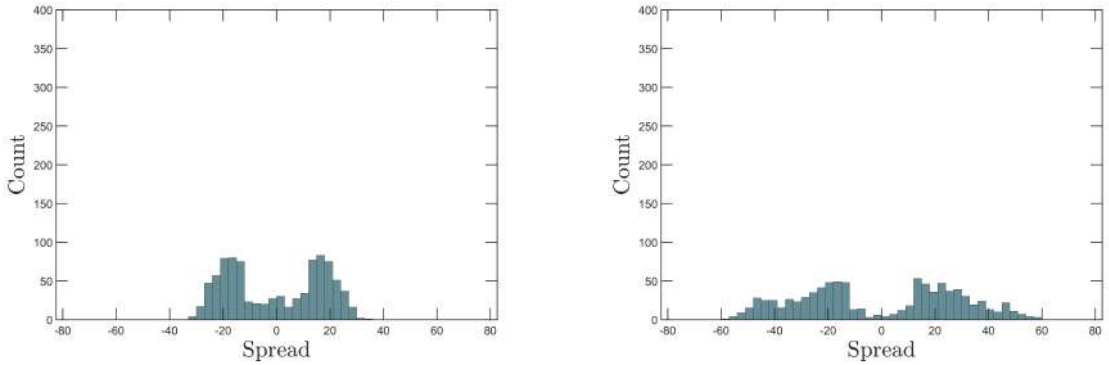
These results led to the conclusion that the internal resonance was indeed a numerical artefact in the conducted simulations. While internal resonance may possibly exist in the system of the OGB, it was not able to be found with Blender during this project.

For a fixed oscillation amplitude, greater oscillation frequency will lead to greater speed and acceleration of the pegs. Despite this, the spread of balls did not appear to be affected by the frequency. The extremes of the distribution were similar for both 10 Hz and 2048 Hz (Figure 61). The simulations showed that beyond a certain oscillation frequency, the spread of the balls reached a maximum. However, the spread of the distribution also appeared to be dependent on the SPS. Higher SPS was found to produce a wider spread distribution for the same oscillation frequency, as shown by Figure 62.



(a) 10 Hz (b) 2048 Hz

Figure 61: Effect of frequency on maximum/minimum spread of distribution (60 SPS)



(a) 60 SPS (b) 120 SPS

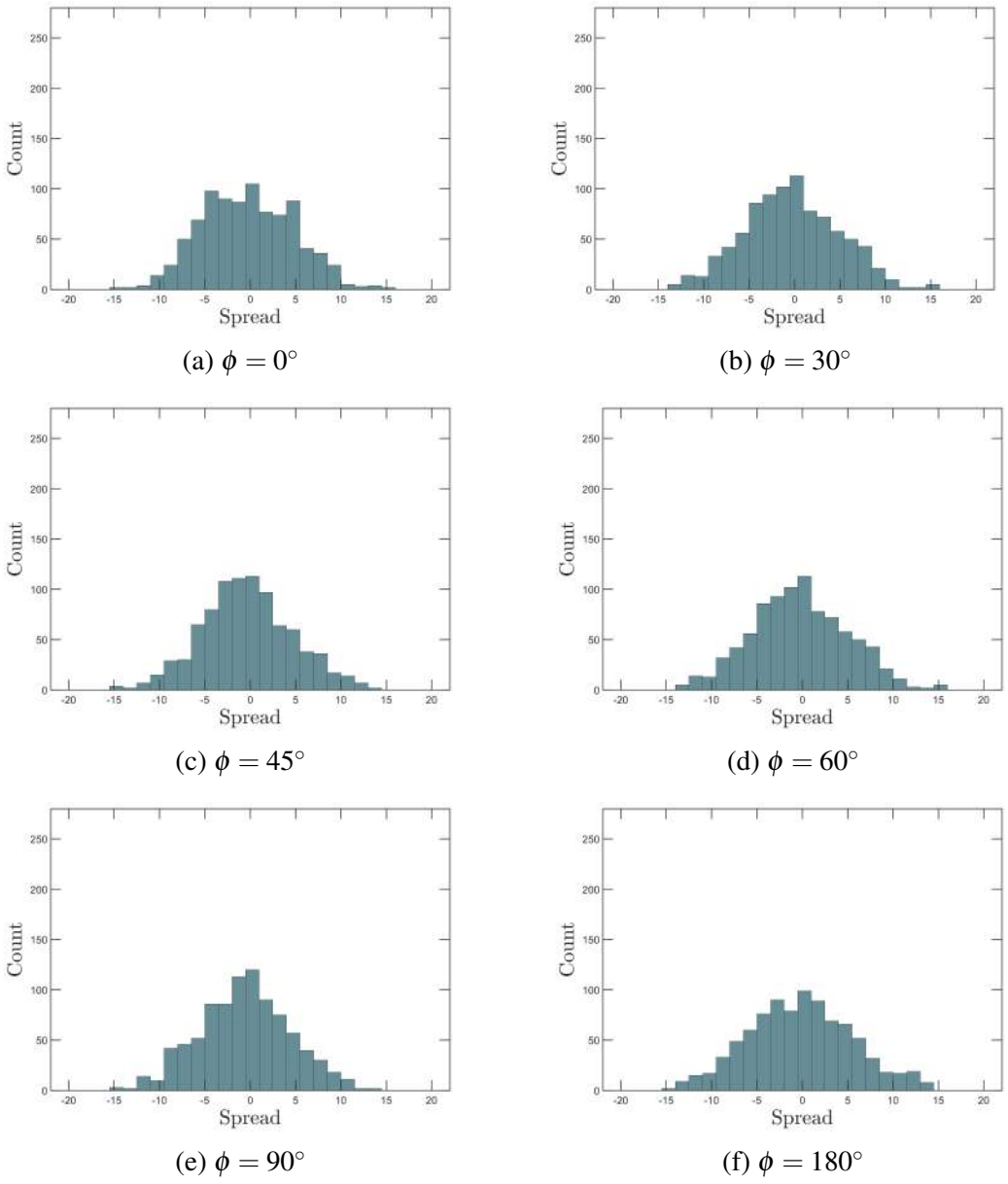
Figure 62: Effect of SPS on spread of distribution (256 Hz)

In November 2020, a new version of Blender was released, version 2.91. In this version, Blender replaced the use of SPS with a new variable, *substeps per frame* (SSPF). With this variable, the number of simulation steps are defined per frame rather than per second (10SSPF at 30 frames/s = 300 SPS). Testing revealed that resonance did not occur with the use of SSPF. This was due to SSPF varying the steps per second with the oscillation frequency, as framerate is defined as 4 times the oscillation frequency, as controlled by the Python script.

Experiments in Blender 2.91 showed that at high frequency, a gap could be formed in the distribution. The width of the gap was able to be controlled by the width of the peg array (Figure 57).

The investigation in Blender was not able to discover internal resonance in the OGB. However, the experimentation revealed key insights into the utility of Blender to investigate dynamic effects. The results indicated that Blender and Bullet are vulnerable to numerical errors and artefacts which may not be easily detected without further experimentation or close scrutiny, as the effects may not be immediately apparent. Future Blender simulations of dynamical systems involving oscillation should take great care to ensure there are no artefacts produced by certain relationships between the SPS or SSPF, FPS and oscillation frequency.

4.3.2 Effect of Phase Difference in the Oscillating Galton Board



(a) $\phi = 0°$ (b) $\phi = 30°$

(c) $\phi = 45°$ (d) $\phi = 60°$

(e) $\phi = 90°$ (f) $\phi = 180°$

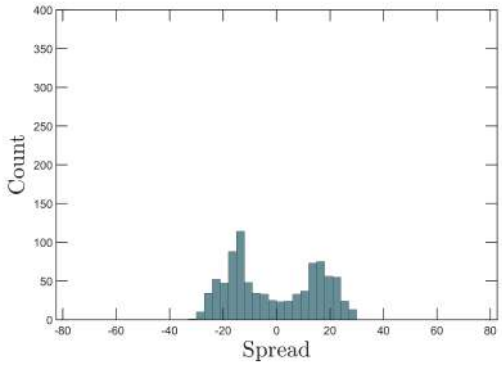
Figure 63: Effects of phase difference ϕ between odd and even rows at 2 Hz

Phase differences of 0°, 30°, 45°, 60°, 90°, 180° were tested for 2 Hz, 24 FPS, 60 SPS. When a phase difference between odd and even rows was applied, the distributions were observed to have a taller and more distinct single peak which was slightly off-centre (skewed). This result was expected.

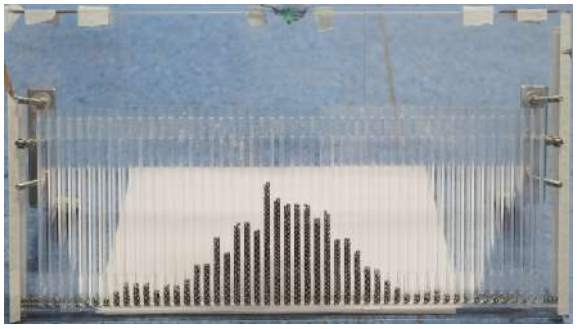
4.3.3 Bullet Simulation vs Mechanical Experiment

A mechanical recreation of the digital OGB experiment was conducted for frequencies of 5, 7, 8, and 10 Hz, each with an amplitude of roughly 3 mm. Slow motion videos were taken

of each experiment. These experiments revealed that for the same amplitude and frequency of oscillation, the mechanical experiment did not replicate the results of the digital experiment.



(a) Digital oscillating Galton board, 10 Hz



(b) Mechanical oscillating Galton board, 10 Hz

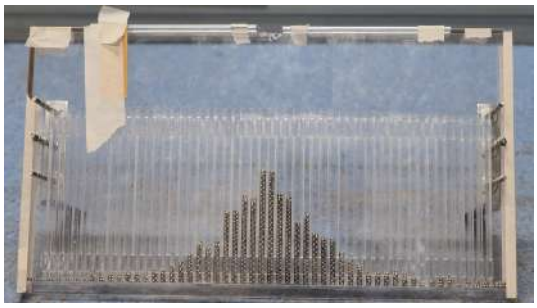
Figure 64: Difference between mechanical and digital results of the oscillating Galton board

In the mechanical experiment, the balls rebounded with smaller velocity after collisions with pegs. As a result, the formation of 2 peaks of the Blender simulations was not found in the distributions of the mechanical experiments. The different trajectories can be seen in the following video https://youtu.be/keTj6uQ3CBE.

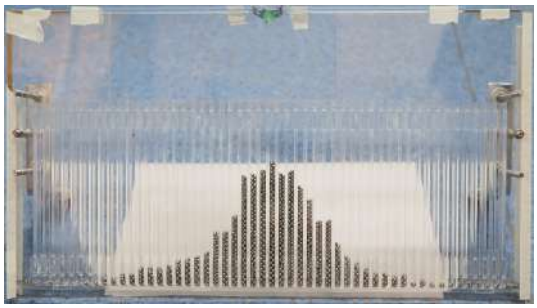
In the digital experiment, the collisions were so elastic such that most of the balls did not pass through the peg array, and were instead thrown to the side of it. However, in the mechanical experiment, most of the balls were able to fall through the peg array, much like they would in the standard static Galton board. As a result, the distribution was largely similar to that of the standard Galton board, which forms a normal distribution. While the mechanical experiment showed that oscillation tends to cause the distribution to broaden, the effect was not as significant as it was in the digital experiment.

In an attempt to reduce the elasticity of the collisions to better mimic the mechanical model, the coefficient of restitution (COR) of the balls and pegs were reduced in the digital model. Mass off the balls were also increased to do reduce the "bounciness" of the collisions. These methods could not produce results that emulate the mechanical experiment. When the COR was set to 0 for both the pegs and balls, there was still significant bouncing in collisions, which showed an error in the simulation. When COR is 0, there should be no observable bouncing.

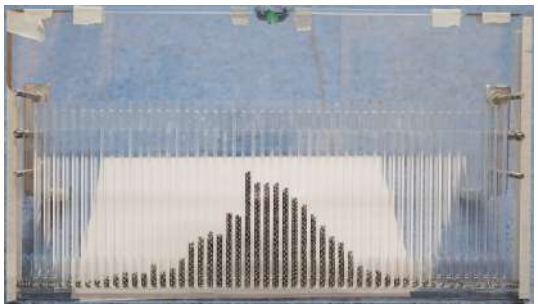
The distributions of the 10 Hz experiment (figs. 65g to 65i) were skewed to the right, which implied that the "filtering" done by the peg array was asymmetric. The 10 Hz experiment was conducted 3 times, which showed very high repeatability of the distribution. In these experiments, the method of pouring the balls into the funnel was changed with little effect to the distribution. Furthermore, the placement of the funnel was also slightly varied with each experiment, as it needed to be removed and replaced with every run of the experiment. Neither of these variables had a significant effect on the distribution.



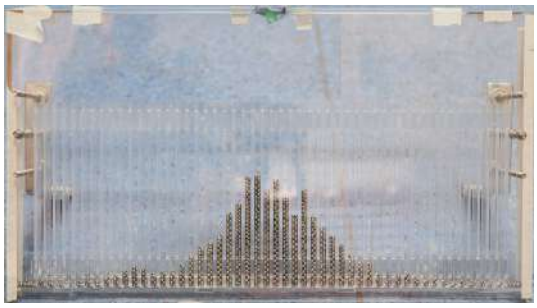
(a) 0 Hz, Static



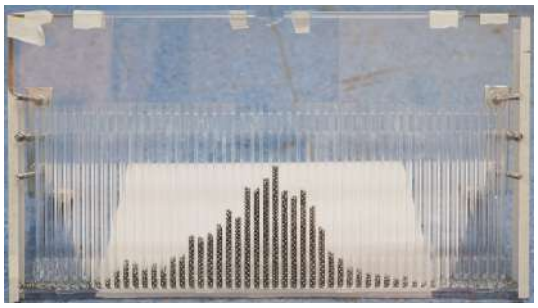
(b) 5 Hz



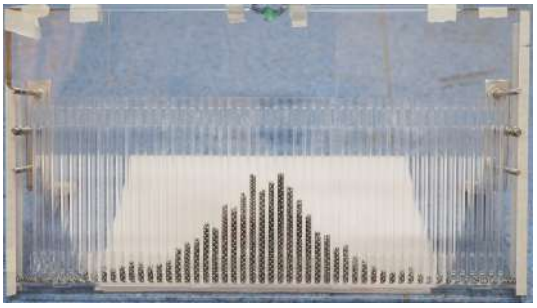
(c) 6 Hz (1 of 2)



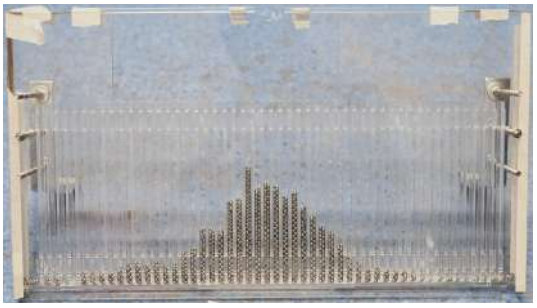
(d) 6 Hz (2 of 2)



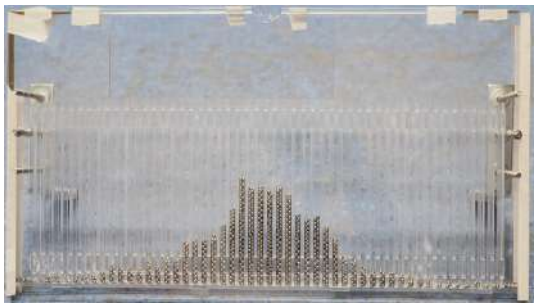
(e) 7 Hz



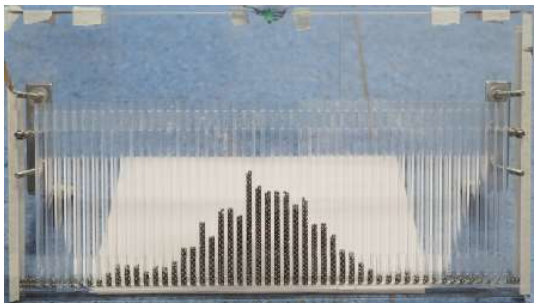
(f) 8 Hz



(g) 10 Hz (1 of 3)



(h) 10 Hz (2 of 3)

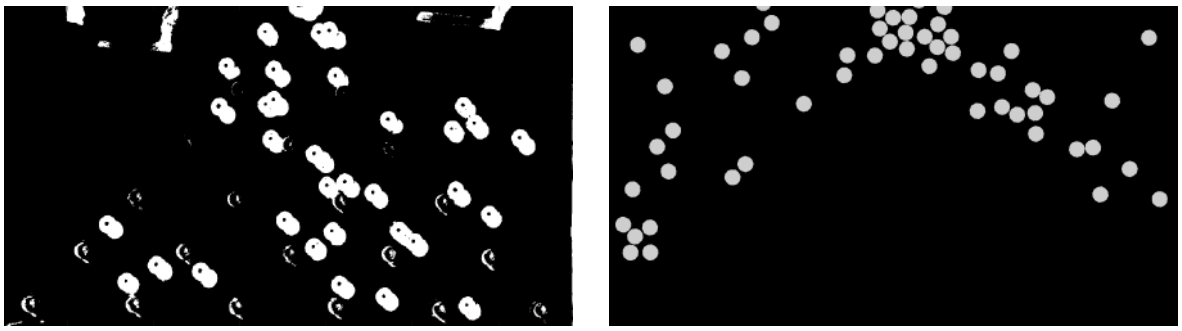


(i) 10 Hz (3 of 3)

Figure 65: Distributions formed by the mechanical oscillating Galton board

The first 6 Hz experiment (Figure 65c) appeared to produce a distribution that was almost identical to those of the 10 Hz experiments. However, this distribution was not repeated by the second 6 Hz experiment (Figure 65d). This would suggest that there may be another factor other than oscillation frequency that is controlling the distribution. This may be the top edges of the bins, which are not perfectly flat due to the fabrication method, and may create some bias when the balls land on top of the bins walls.

A slow-motion video-recording was taken of each of the mechanical experiments. The recording of the 10 Hz experiment was analysed in Matlab with image processing to directly compare the mechanical and digital results numerically with the method outlined in section 3.



(a) Mechanical, "background" removed (b) Digital

Figure 66: Processed frames from slow-motion video of mechanical and digital experiments

The average entropy of the mechanical model (0.3583) was higher than that of the digital model (0.279565) due to more balls passing through the peg array. As a benchmark, the entropy of gaussian noise was also calculated to be 3.8079. The Kruskal-Wallis test returned a *p-value* of $3.28e{-}266$, which indicated a negligible probability that the 2 sets of data (digital and mechanical) came from the same distribution.

While the mean values of the entropies are close in value, a high fidelity simulator that involves the fundamental and differential equations is needed to be able to draw a conclusion regarding the accuracy of the models. The numerical comparison of the entropies is far less significant than the visually observed collisions and motion.

The difference between the mechanical and digital experimental results could have been due to inaccuracies caused by Bullet's approximation of contact force as constant over the entire contact duration. The law of conservation of momentum is crucial to the accuracy of collisions such as those in the oscillating Galton board. This law determines the final velocity of colliding objects, and asserts that when a peg collides with a ball, the velocity of the peg should decrease. However, in the digital model, it was impossible for the pegs to be moved by the the balls, as the motion of the pegs is "locked" by keyframes. In other words, the motion path of the pegs is fixed, which leads to inconsistencies between the simulated and actual collisions.

Furthermore, due to limitations of Bullet and Blender, the scale of the digital experiment was different. While the mechanical balls were 3 mm in diameter, the digital balls had to be

3 m in diameter, as smaller scale dynamics caused extreme artefacts, such as the balls moving through the other objects without colliding. Using metre scale with a scaled-up gravitational constant, 9810 m/s^2, led to the same artefacts as the millimetre scale with the standard 9.81 m/s^2. As a result of the different scale of the digital model, at a given frequency, the vertical speed of the balls is slower relative to the horizontal speed of the pegs, compared to the mechanical model.

The mechanical OGB experiments showed that the short-cuts taken by Bullet to increase computational speed led to inaccuracies in the simulation of collisions. However, the experiments revealed that the stochastic input can still produce a predictable and repeatable result in the Galton board, even when the pegs are made to oscillate. In the Bullet simulations, the splitting of the distribution to form 2 peaks did not occur until frequency was sufficiently high. The mechanical model will likely produce the splitting when the oscillation frequency is sufficiently high. However, this could not be done with the available apparatus as the LDS electrodynamic shaker used was not able to produce oscillations greater than 10 Hz with amplitudes of 3 mm.

During this project, the implementation of scripting with Python code in Blender allowed experiments to be conducted autonomously. The user can use the code to define which experiments to conduct, what variables to change, and which results to record. Variables such as magnetic field strength and oscillation frequency can be set as variables in the Python code. The throughput of these tests scales with the number of CPU cores or machines, which allow multiple simulations to be run simultaneously. These factors make Blender well suited to experiments that require many repeated near-identical tests, such as the oscillating Galton board. However, Blender's primary interface is its graphical interface, not the text interface that implements code. As such, not every function in Blender can be conducted through code.

5 Conclusion and Outlook

5.1 Investigation of Nonlinear Dynamic Systems

The rising importance of nonlinear dynamics in engineering, such as in floating offshore wind turbines to achieve renewable energy goals, has created a need for an experiential approach to introduce these mechanics to undergraduate engineering students.

Therefore, this project aimed to develop nonlinear systems in Blender to investigate nonlinear phenomena and chaotic behaviour. This was achieved through the simulation of nonlinear chaotic models from literature, which showed that chaotic phenomena could be detected using the same methods from the graphical analysis of real systems in engineering. This analysis revealed that the Blender simulations showed the markers of chaos that were suggested by theory and experimental results from literature. Moreover, the visual nature of the analysis will lend itself well to an educational laboratory.

The oscillating Galton board (OGB) was simulated to investigate nonlinearities in dynam-

ical systems with statistical outputs. Through experimentation by simulation in Blender, the research questions concerning the OGB could be answered. The simulations showed that as the oscillation frequency of the pegs was increased, the distribution tended to split to form 2 roughly symmetrical peaks on either side of the centre. However, this could not be replicated in the mechanical model due to the constrained frequency range allowed by the equipment.

When phase difference was applied between the oscillations of odd and even rows of pegs, the distribution narrowed and became skewed to either the left or right. While there was an observed cyclically occurring internal resonance in the OGB, it was found to be a numerical error.

The mechanical OGB showed that the distributions were still highly predictable and repeatable with the oscillating pegs, as they were in the static Galton board.

5.2 Blender in Digital Experimentation and Pedagogy

One of the primary objectives of this project was to establish if Blender could adequately capture nonlinear phenomena to create insights to Blender's simulations and potential contribution to pedagogy. Over the course of this project, Blender and Bullet were found to be a powerful tool for generating fast and accurate simulations of nonlinear systems. The intuitive Blender interface and fast Bullet engine encouraged experimentation of the dynamical systems.

Graphical markers of chaos detected in simulations showed that Blender was able to replicate the expected experimental and theoretical results. The same methods to detect chaos in real engineering systems were successfully used to detect markers of chaos in the digital experiments. In doing so, the digital laboratory performed the same function as a conventional laboratory, in that it was able to make the user reproduce theoretical results through an experiential activity. Furthermore, the visual nature of the analysis with Poincaré maps, phase planes and Fourier spectra should aid significantly to the retention and attention of the student's learning experience.

In the past, there have been concerns that digital laboratories lack sufficient realism without 3D graphics. As Blender is primarily an animation tool, it has powerful tools for creating photo-realistic 3D graphics. The rendering engine, *Eevee*, is able to render realistic surfaces in real time in Blender, which could aid in creating a sense of realism.

Digital laboratories have also been criticised for their lack of interactivity compared to real labs. While Blender and Bullet cannot completely overcome this, the fast and highly simplified implementation of simulation greatly encourages interactivity and experimentation compared to conventional simulation packages, which often require many more parameters to be set before a simulation can be conducted. Blender's potential for digital laboratories could be greatly improved when its implementation of VR has been more thoroughly developed.

This investigation of Bullet and Blender opens up options for digital experiments during future pandemics or natural disasters that might inhibit students from attending conventional

laboratories.

The methods of this project can be used as a blueprint for future such laboratories. In UCD, these simulation experiments can be implemented as a laboratory or coursework in MEEN30010 Applied Dynamics II or MEEN41040 Advanced Vibrations to introduce undergraduate engineering students to the concepts of nonlinearity and chaos.

5.3 Limitations

One of Blender's greatest strengths is the high speed physics engine, Bullet. However, Bullet is not without compromise. To achieve the high computational speed, sacrifices had to be made in the accuracy of the simulations.

Limitations to Bullet and Blender's simulation were discovered during this project. Bullet was found to be inaccurate in the simulation of frictionless dynamic contact between 2 rigid bodies. In addition to this, the hinge constraint in Blender was found to be imperfect, as it had play and hence introduced unintended nonlinearity to the system.

The oscillating Galton board simulations showed inaccurate collisions when compared to a mechanical counterpart. The coefficient of restitution implementated in Bullet was also showed to be inaccurate, as when it was set to 0, there was still bouncing in collisions.

These errors limit Blender's capability in accurately simulating dynamical systems, and demonstrated Bullet's shortcomings.

In the mechanical oscillating Galton board model, the LDS electrodynamic shaker could not provide oscillations greater than 10 Hz with the desired amplitude of 3 mm. Therefore, the splitting of the distribution could not be reproduced in the mechanical experiment.

While Blender is able to achieve high speed simulation with Bullet, it is still unable to provide the instant response and real-time feedback that a mechanical system provides. This limits the interactivity of Blender, as the response of a dynamical system to a change in variable will take more time to compute than it would to occur in a real system.

5.4 Future Work

In UCD, the findings of this investigation will have a direct effect on how nonlinearities can be introduced to undergraduate mechanical engineers in the future, such as in the courses, MEEN30010 Applied Dynamics II or MEEN41040 Advanced Vibrations.

While the Blender interface is much simpler than conventional dynamical simulation packages, its learning curve is still very steep relative to general software. Despite this, the set-up of the dynamic systems is quick and easy if the steps are provided to the user. Hence, with a provided step-by-step guide, which is common for laboratories, a completely new user will be able to create a simulation of a nonlinear dynamic system and begin experimentation on it. The author's recommendation is that these instructions should be specific in terms of what settings and objects to click on, create or enable, but details such as the length of the pendulum,

weight or magnetic strength should be left up to the student. The student should not feel as though they are recreating a precise system that only produces the desired result when all of the settings are exactly right. By leaving such settings up to the student's discretion, the student will be able to get a better grasp on how chaos can arise from almost any amount of nonlinearity.

Further investigation should be conducted to study the reason behind the difference between the collisions of the mechanical and digital oscillating Galton board models. Future studies could focus on establishing the limitations and accuracy of these collisions.

The use of VR in Blender is currently limited to only viewing the scene and objects cannot be moved or interacted with. As the implementation of VR becomes developed further, future studies should investigate its use in digital laboratories for education in engineering.

Dissemination

Project supervisor Pakrashi is preparing 2 journal papers involving the discoveries of this project for the journals, *Physical Review* and *Nonlinear Dynamics*.

A proposal is also being written to present the findings at the *10^{th} European Nonlinear Dynamics Conference (ENOC) 2021*, hosted by University of Lyon.

The results of the observed internal resonance in the oscillating Galton board were presented at the *Internal Conference on Engineering Vibration 2020*, hosted by University of Aberdeen on 15 December 2020.

For the video competition, *This is Engineering 2021*, held by National University of Ireland Galway in collaboration with Engineers Ireland on 16 March 2021, the author created a video to inform 5^{th} and 6^{th} year secondary school students on what mechanical engineering involves, which briefly included content from this project. This video won second prize and can be found at https://youtu.be/aNEc1L5WnsA.

The overall results and progress of the project were presented at a poster session at *Work Smarter Together 2021*, hosted by University College Dublin on 8 March 2021. These were also presented at *All Ireland Conference on Undergraduate Research 2021*, hosted by University of Limerick on 24 March 2021.

References

[1] Ashraf Zaher. Digital Communication using a Novel Combination of Chaotic Shift Keying and Duffing Oscillators. *International journal of innovative computing, information & control: IJICIC*, 9:1865–1879, May 2013.

[2] Francis C. Moon. *Chaotic and Fractal Dynamics: Introduction for Applied Scientists and Engineers*. John Wiley & Sons, November 2008.

[3] Eoghan Phelan. Developing Dynamic Toys: Investigating a Biased Galton Board and the Brachistochrone problem, April 2020.

[4] Setting the Viewpoint with Azimuth and Elevation - MATLAB & Simulink - MathWorks United Kingdom.

[5] Ahmet Özer and Erhan Akin. TOOLS FOR DETECTING CHAOS. *Sakarya University Journal of Science*, 9(1):60–66, June 2005. Number: 1.

[6] Jianlian Cheng and Hui Xu. Nonlinear dynamic characteristics of a vibro-impact system under harmonic excitation. *Journal of Mechanics of Materials and Structures - J MECH MATER STRUCT*, 1:239–258, June 2006.

[7] Simón Reif-Acherman. Toys as teaching tools in engineering: the case of Slinky®. 17:111–122, July 2015.

[8] Anna Phillips, Paul Palazolo, Susan Magun-Jackson, Charles Camp, and Douglas Schmucker. "Powerful Play: Using Toys As Tools In Engineering Education". pages 7.5.1–7.5.16, June 2002. ISSN: 2153-5965.

[9] J. Fernandez de Canete, C. Galindo, and Inmaculada Garcia-Moral. *System Engineering and Automation: An Interactive Educational Approach*. Springer-Verlag, Berlin Heidelberg, 2011.

[10] Avi Hofstein. The Role of Laboratory in Science Teaching and Learning. pages 357–368. January 2017.

[11] Lyle Feisel and A.J. Rosa. The Role of the Laboratory in Undergraduate Engineering Education. *Journal of Engineering Education*, 94, January 2005.

[12] Balamuralithara Balakrishnan and Peter Woods. Virtual laboratories in engineering education: The simulation lab and remote lab. *Computer Applications in Engineering Education*, 17:108–118, March 2009.

[13] Blender Game Engine Physics Introduction. docs.blender.org/manual/en/2.79/game_engine/physics.

[14] Hantao He, Junxing Zheng, Quan Sun, and Zhaochao Li. Simulation of Realistic Particles with Bullet Physics Engine. 92:14004, June 2019. Conference Name: E3S Web of Conferences.

[15] Ehsan Izadi and Adam Bezuijen. Simulating direct shear tests with the Bullet physics library: A validation study. *PLOS ONE*, 13(4):e0195073, April 2018. Publisher: Public Library of Science.

[16] Ned J. Corron. Chaos in a linear wave equation. *Chaos, Solitons & Fractals: X*, 2:100014, June 2019.

[17] Jens Feder. *Fractals*. Physics of Solids and Liquids. Springer US, 1988.

[18] Wai Kai Chen. *The Electrical Engineering Handbook*. Elsevier, November 2004. Google-Books-ID: qhHsSlazGrQC.

[19] Mohd Shakir Md Saat. Analysis and Synthesis of Polynomial Discrete-Time Systems - 1st Edition, July 2017.

[20] Anindya Chatterjee. A Brief Introduction to Nonlinear Vibrations. page 20, February 2009.

[21] David Ruelle. Strange Attractors. In *Turbulence, Strange Attractors and Chaos*, volume Volume 16 of *World Scientific Series on Nonlinear Science Series A*, pages 195–206. WORLD SCIENTIFIC, September 1995.

[22] Xinxing Wu and Peiyong Zhu. Chaos in a class of non-autonomous discrete systems. *Applied Mathematics Letters*, 26(4):431–436, April 2013.

[23] Daniel Toker, Friedrich T. Sommer, and Mark D'Esposito. A simple method for detecting chaos in nature. *Communications Biology*, 3(1):1–13, January 2020. Number: 1 Publisher: Nature Publishing Group.

[24] Ying-Cheng Lai and Tamás Tél. *Transient Chaos: Complex Dynamics on Finite Time Scales*. Applied Mathematical Sciences. Springer-Verlag, New York, 2011.

[25] Chongxin Liu, Ling Liu, and Tao Liu. A novel three-dimensional autonomous chaos system. *Chaos, Solitons & Fractals*, 39:1950–1958, February 2009.

[26] Yuxin Xie and Siming Liu. From period to quasiperiod to chaos: A continuous spectrum of orbits of charged particles trapped in a dipole magnetic field. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(12):123108, December 2020. Publisher: American Institute of Physics.

[27] Ray Brown and Leon O. Chua. Clarifying chaos: examples and counterexamples. *International Journal of Bifurcation and Chaos*, 06(02):219–249, February 1996. Publisher: World Scientific Publishing Co.

[28] Colin Y. Shen, Thomas E. Evans, and Steven Finette. Polynomial chaos quantification of the growth of uncertainty investigated with a Lorenz model. *Journal of Atmospheric and Oceanic Technology*, 27(6):1059–1071, 2010. Number: 6.

[29] Daniel Schertzer and Shaun Lovejoy. Uncertainty and predictability in geophysics: Chaos and multifractal insights. In Robert Stephen John Sparks and Christopher John Hawkesworth, editors, *Geophysical Monograph Series*, volume 150, pages 317–334. American Geophysical Union, Washington, D. C., 2004.

[30] A. L. Goldberger, D. R. Rigney, and B. J. West. Chaos and fractals in human physiology. *Scientific American*, 262(2):42–49, February 1990.

[31] Mark F. Bocko, David H. Douglass, and Heidi H. Frutchy. Bounded regions of chaotic behavior in the control parameter space of a driven non-linear resonator. *Physics Letters A*, 104(8):388–390, September 1984.

[32] A. P. Kuznetsov, S. P. Kuznetsov, and N. V. Stankevich. A simple autonomous quasiperiodic self-oscillator. *Communications in Nonlinear Science and Numerical Simulation*, 15(6):1676–1681, June 2010.

[33] L. Borkowski and A. Stefanski. FFT Bifurcation Analysis of Routes to Chaos via Quasiperiodic Solutions, December 2015. ISSN: 1024-123X Pages: e367036 Publisher: Hindawi Volume: 2015.

[34] James Theiler. Estimating fractal dimension. *JOSA A*, 7(6):1055–1073, June 1990. Publisher: Optical Society of America.

[35] Hendrik Wernecke, Bulcsú Sándor, and Claudius Gros. How to test for partially predictable chaos. *Scientific Reports*, 7(1):1087, April 2017. Number: 1 Publisher: Nature Publishing Group.

[36] Find Periodicity Using Autocorrelation - MATLAB & Simulink - MathWorks United Kingdom.

[37] Bingbing Qiu, Guofeng Wang, Yunsheng Fan, Dongdong Mu, and Xiaojie Sun. Adaptive Sliding Mode Trajectory Tracking Control for Unmanned Surface Vehicle with Modeling Uncertainties and Input Saturation. *Applied Sciences*, 9(6):1240, January 2019. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[38] Jiří Náprstek and Cyril Fischer. Stability of limit cycles in autonomous nonlinear systems. *Meccanica*, 49(8):1929–1943, August 2014.

[39] Jae Young Lee, Nam Seok Kim, and Mamoru Ishii. Flow regime identification using chaotic characteristics of two-phase flow. *Nuclear Engineering and Design*, 238(4):945–957, April 2008.

[40] Han L. J. van der Maas, Paul F. M. J. Verschure, and Peter C. M. Molenaar. A note on chaotic behavior in simple neural networks. *Neural Networks*, 3(1):119–122, January 1990.

[41] P. Bergé. Study of the Phase Space Diagrams Through Experimental Poincaré Sections in Prechaotic and Chaotic Regimes. *Physica Scripta*, T1:71–72, January 1982. Publisher: IOP Publishing.

[42] F. C. Moon and W. T. Holmes. Double Poincaré sections of a quasi-periodically forced, chaotic attractor. *Physics Letters A*, 111(4):157–160, September 1985.

[43] Katja Mombaur, Heike Vallery, Yue Hu, Jonas Buchli, Pranav Bhounsule, Thiago Boaventura, Patrick M. Wensing, Shai Revzen, Aaron D. Ames, Ioannis Poulakakis, and Auke Ijspeert. Chapter 4 - Control of Motion and Compliance. In Maziar A. Sharbafi and André Seyfarth, editors, *Bioinspired Legged Locomotion*, pages 135–346. Butterworth-Heinemann, January 2017.

[44] Celso Grebogi, Edward Ott, and James A. Yorke. Chaos, Strange Attractors, and Fractal Basin Boundaries in Nonlinear Dynamics. *Science*, 238(4827):632–638, October 1987. Publisher: American Association for the Advancement of Science Section: Articles.

[45] Bernard Brogliato, Rogelio Lozano, Bernhard Maschke, and Olav Egeland. *Dissipative Systems Analysis and Control: Theory and Applications*. Communications and Control Engineering. Springer-Verlag, London, 2 edition, 2007.

[46] Naofumi Tsukamoto and Hirokazu Fujisaka. Return map analysis of chaotic phase synchronization. *Physica D: Nonlinear Phenomena*, 233(1):32–38, September 2007.

[47] Hiromichi Suetani, Karin Soejima, Rei Matsuoka, Ulrich Parlitz, and Hiroki Hata. Manifold learning approach for chaos in the dripping faucet. *Physical Review E*, 86(3):036209, September 2012. Publisher: American Physical Society.

[48] Tomasz Kapitaniak. *Chaos for Engineers: Theory, Applications, and Control*. Springer-Verlag, Berlin Heidelberg, 2 edition, 2000.

[49] Evgeni V. Nikolaev, Sahand Jamal Rahi, and Eduardo D. Sontag. Subharmonics and Chaos in Simple Periodically Forced Biomolecular Models. *Biophysical Journal*, 114(5):1232–1240, March 2018.

[50] Sven Erik Jørgensen. Encyclopedia of Ecology. *Annals of Botany*, 105(3):335–346, March 2010.

[51] D. W. Jordan and Peter Smith. *Nonlinear ordinary differential equations: an introduction for scientists and engineers*. New York : Oxford University Press, Oxford [England], 4th ed edition, 2007.

[52] Ludomir Oteski, Yohann Duguet, Luc Pastur, and Patrick Le Quéré. Quasiperiodic routes to chaos in confined two-dimensional differential convection. *Physical Review E*, 92(4):043020, October 2015. Publisher: American Physical Society.

[53] Toshiki Aikawa. The Pomeau-Manneville intermittent transition to chaos in hydrodynamic pulsation models. *Astrophysics and Space Science*, 139(2):281–293, December 1987.

[54] Alan Wolf, Jack B. Swift, Harry L. Swinney, and John A. Vastano. Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285–317, July 1985.

[55] Judith R. Brown and Michael B. McGrath. Visual learning for science and engineering. *IEEE computer graphics and applications*, 25(5):56–63, October 2005.

[56] Sang Gyu Kwak and Jong Hae Kim. Central limit theorem: the cornerstone of modern statistics. *Korean Journal of Anesthesiology*, 70(2):144, 2017.

[57] V. V. Kozlov and M. Yu Mitrofanova. Galton Board. *arXiv:nlin/0503024*, March 2005. arXiv: nlin/0503024.

[58] Bill Moran, William G. Hoover, and Stronzo Bestiale. Diffusion in a periodic Lorentz gas. *Journal of Statistical Physics*, 48(3):709–726, August 1987.

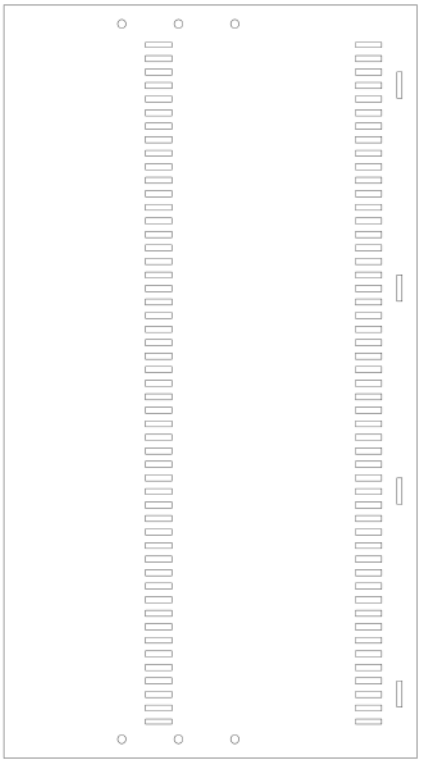
[59] Office Playground. Space Balls - Perpetual Motion, October 2010.

[60] Fraser. Chaos and detection. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 1996.

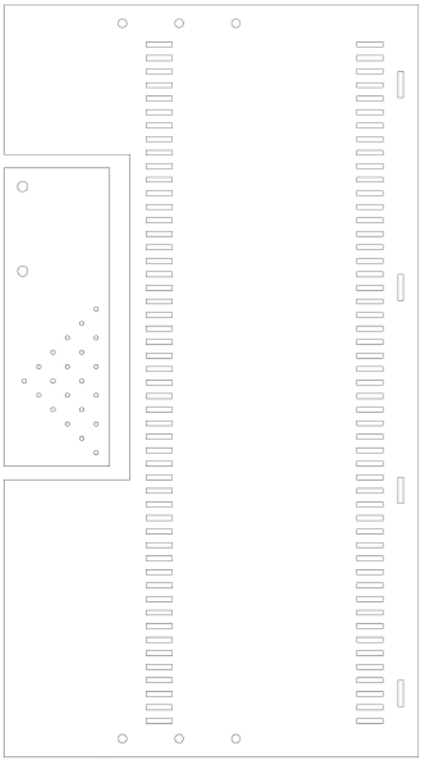
[61] Gregory Baker. Probability, pendulums, and pedagogy. *American Journal of Physics - AMER J PHYS*, 74, June 2006.

[62] Ang Zhou, Wang Shilian, and Junshan Luo. Blind Frequency Estimation and Symbol Recovery for the Analytically Solvable Chaotic System. *Entropy*, 21:791, August 2019.

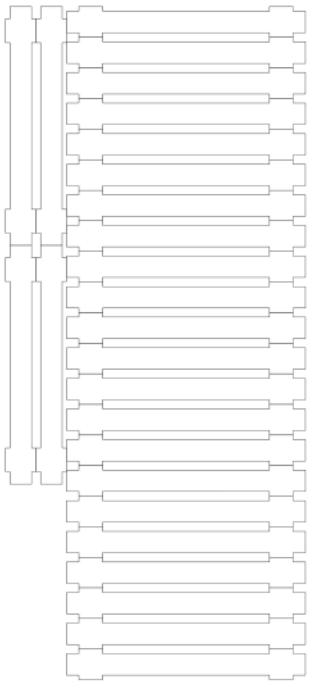
6 Appendix



(a) Front (445×245 mm)



(b) Back (445×245 mm)



(c) Bin walls (156×20 mm, each)

Figure 67: Drawings of laser-cut parts for mechanical oscillating Galton board

The following script generates the oscillating Galton board when it is run in the Blender Text Editor. The user must define parameters such as oscillation frequency, oscillation amplitude or number of pegs. This version of the script is a modification of code written by Eoghan Phelan.

```
# Authors: Eoghan Phelan, Aidan Lee
# This script is must be run in the Blender Text Editor.
# This script generates the oscillating Galton board model, runs the simulation, and exports the result

import bpy
import os
import csv
from math import sin

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)  # Deletes previous configuration before creating again
bpy.ops.ptcache.free_bake_all() # delete previous bake


context = bpy.context
scene = context.scene

# Go to frame 0
bpy.context.scene.frame_set(0)

# define the first and last frequencies to be tested
# NOTE: DO NOT CHANGE lastf. TEST 1 FREQUENCY AT A TIME. THE LOOP NO LONGER WORKS IN v2.91.
firstf=10
lastf=firstf+1

# MORE INPUTS ARE REQUIRED FROM LINE 65 TO LINE 92
# DIRECTORY FOR SIMULATION RESULT MUST BE DEFINED ON LINE 307

# define the substeps per frame
sspf=5

for f in range(firstf,lastf):

    # SET BAKE SETTINGS

    # Set framerate
    if f<6:
        FPS=24
    else:
```

```
        FPS=4*f # framerate of animation

    bpy.context.scene.render.fps=FPS

    # Set Frame Start and End
    bpy.context.scene.frame_start=0

    endframe=(FPS/24)*10000
    bpy.context.scene.frame_end=endframe

    # Go to frame 0
    bpy.context.scene.frame_set(0)
    bpy.ops.object.select_all(action='SELECT')
    bpy.ops.object.delete(use_global=False) # Deletes previous
   configuration before creating again

    bpy.ops.ptcache.free_bake_all() # delete previous bake


    context = bpy.context
    scene = context.scene

    #####

    for c in scene.collection.children:
        scene.collection.children.unlink(c)

    for c in bpy.data.collections:
        if not c.users:
            bpy.data.collections.remove(c)        # Creates scene

    # INPUTS FOR SINUSOIDAL OSCILLATION OF PEGS
    pi=3.14159265358979323846264338327950288419716939937510

    A=0.325 # input the amplitude of oscillation
    f=f # input the frequency of oscillation

    phase_diff=0 # input the offset of oscillation

    # INPUTS FOR GALTON BOARD DIMENSIONS & PARAMETERS
    s = 0.25  # Diameter of the scater pegs

    rgapx = 0.84855 # Distance between pegs to the right in x direction

    gapz =rgapx # Distance between pegs in z direction

    lgapx =rgapx # Distance between pegs to the left in x direction
```

```
86
87      n = 6  # Number of rows of pegs
88      slotn = 600 # Number of bins_
89
90      cube = 0.3 # Diameter of particle
91
92      cubex = 40 # Number of x rows of particles
93      cubez = 25  # Number of z rows of particles
94      cubemass=0.0002
95
96      ramp_offset=1.4*cube # control the size of the funnel opening
97      extra=200 # spread of bins
98
99      # CALCULATION OF CONSTANTS
100     if f==0:
101         period=1
102     else:
103         period=FPS/f
104
105     l = n*gapz*4.75*2.5 # length of the board
106     w = (n*lgapx + rgapx*n+extra) # width of the bins
107     w2 = (n*lgapx + rgapx*n)*1.5 # width of back board
108     bret = cube*1.05 # breadth of the board
109
110     ramp = l # length of funnel
111
112     x = 0               # Default x position
113     y = -bret*0.5       # Default y position
114     z = l*0.6           # Default z position
115
116     # PLACE CAMERA AND LIGHT
117     bpy.ops.object.camera_add(enter_editmode=False, align='VIEW',
    location=(0,72.404, 25.0051),
118         rotation=(90/57.3, 0, 180/57.3), scale=(1, 1, 1))
119     bpy.ops.object.light_add(type='SUN', location=(0, 180.081, 188.388),
120         rotation=(-90/57.3, 0, 0), scale=(1, 1, 1))
121
122
123     # CONTRUCTION OF OBJECTS
124
125     bin_collection = bpy.data.collections.new('bins')
126     bpy.context.scene.collection.children.link(bin_collection) # Creates
     collection of the bins
127
128     # Construction of the the back board
129     bpy.ops.mesh.primitive_cube_add(size=1, location=(0 , bret*0.5, l
    *0.5))
```

```
130     bpy.ops.transform.resize(value=(extra+w2, bret, l*2),
    constraint_axis=(True, True, True))
131     bpy.ops.rigidbody.object_add()
132     bpy.context.object.rigid_body.type = 'PASSIVE'
133     bpy.context.object.rigid_body.use_margin = True
134     bpy.context.object.rigid_body.collision_margin = 0.0001
135     bpy.context.object.hide_viewport = False
136     bpy.context.object.hide_render = True
137 
138     # Construction of the front board
139     bpy.ops.mesh.primitive_cube_add(size=1, location=(0 , -bret*1.5, l
    *0.5))
140     bpy.ops.transform.resize(value=(extra+w2, bret, l*2),
    constraint_axis=(True, True, True))
141     bpy.ops.rigidbody.object_add()
142     bpy.context.object.rigid_body.type = 'PASSIVE'
143     bpy.context.object.rigid_body.use_margin = True
144     bpy.context.object.rigid_body.collision_margin = 0.0001
145     bpy.context.object.hide_viewport = False
146     bpy.context.object.hide_render = True
147 
148 
149 
150     # Construction of the floor board
151     bpy.ops.mesh.primitive_cube_add(size=1, location=(0, 0, -bret*0.5))
152     bpy.ops.transform.resize(value=(extra+w2, bret*4, bret),
    constraint_axis=(True, True, True))
153     bpy.ops.rigidbody.object_add()
154     bpy.context.object.rigid_body.type = 'PASSIVE'
155     bpy.context.object.rigid_body.use_margin = True
156     bpy.context.object.rigid_body.collision_margin = 0.0001
157 
158     #Code that constucts the 1st ramp
159 
160     bpy.ops.mesh.primitive_cube_add(size=1, location=(x - 1.5*cube -
    0.5*ramp*0.72 + ramp_offset,
161         y, z + 1.5*gapz + ramp*0.5*0.72))
162     bpy.ops.transform.resize(value=(ramp, bret, s), constraint_axis=(
    True, True, True))
163     bpy.ops.transform.rotate(value=0.7853981, orient_axis='Y',
    orient_type='GLOBAL',
164         orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0, 1)),
165         orient_matrix_type='GLOBAL', constraint_axis=(True, True, True)
    ,)
166     bpy.ops.rigidbody.object_add()
167     bpy.context.object.rigid_body.type = 'PASSIVE'
168     bpy.context.object.rigid_body.use_margin = True
```

```
169     bpy.context.object.rigid_body.collision_margin = 0.0001
170     bpy.context.object.rigid_body.friction = 0
171 
172     #Code that constucts the 2nd ramp
173     bpy.ops.mesh.primitive_cube_add(size=1, location=(x + 1.5*cube +
    0.5*ramp*0.72 - ramp_offset,
174         y, z + 1.5*gapz + ramp*0.5*0.72))
175     bpy.ops.transform.resize(value=(ramp, bret, s), constraint_axis=(
    True, True, True))
176     bpy.ops.transform.rotate(value=-0.7853981, orient_axis='Y',
    orient_type='GLOBAL',
177         orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0, 1)),
178         orient_matrix_type='GLOBAL', constraint_axis=(True, True, True)
    ,)
179     bpy.ops.rigidbody.object_add()
180     bpy.context.object.rigid_body.type = 'PASSIVE'
181     bpy.context.object.rigid_body.use_margin = True
182     bpy.context.object.rigid_body.collision_margin = 0.0001
183     bpy.context.object.rigid_body.friction = 0
184 
185 
186     # CONSTRUCTION OF OBJECTS IN LOOPS
187 
188     count = 0
189 
190     # Loop that contstructs the specified number of scatter pegs
191     while (count < n):
192 
193 
194 
195         for i in range (0, n-count):
196             bpy.ops.mesh.primitive_cylinder_add(radius=s*0.5, depth=bret
    , enter_editmode=False,
197                 location=((x - lgapx*count) + rgapx*i, y, (z - gapz*
    count) - gapz*i))
198             bpy.ops.transform.rotate(value=2*0.7853981, orient_axis='X',
     orient_type='GLOBAL',
199                 orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0, 1)),
200                 orient_matrix_type='GLOBAL', constraint_axis=(True, True
    , True),)
201             bpy.ops.rigidbody.object_add()
202             bpy.context.object.rigid_body.type = 'PASSIVE'
203             bpy.context.object.rigid_body.use_margin = True
204             bpy.context.object.rigid_body.collision_margin = 0.0001
205             bpy.context.object.rigid_body.friction = 0.25
206             bpy.context.object.rigid_body.restitution = 0.25
207 
```

```
208             peg = bpy.context.selected_objects[0] # newly created peg
    will be automatically selected
209             bpy.context.object.rigid_body.kinematic = True # make it
    animated/kinematic
210             Peg=bpy.context.object
211
212             # Create keyframes of the peg movements
213
214             dot=0
215             x0=Peg.location[0]
216
217             if f!=0:
218                 if (i%2==0 and count%2==0) or (i%2!=0 and count%2!=0):
219                     C=phase_diff
220                 else:
221                     C=0
222
223                 while (dot<period+1):
224
225                     # calculate displacement for a frame
226                     x=A*sin(2*pi*(f/FPS)*dot+C)
227
228                     Peg.location[0] = x0+x # location is its original
    position + the displacement
229                     Peg.keyframe_insert(data_path="location", frame=dot)
     # Set the keyframe with that location, and which frame.
230
231                     dot+=1
232
233                 #repeat the movement infinitely with cycles modifier
234                 obj = bpy.context.selected_objects[0]
235                 #pick the fcurve to add to
236                 fcurve = obj.animation_data.action.fcurves[0]
237                 #add a modifier and get a reference.
238                 fcurve = fcurve.modifiers.new(type='CYCLES')
239         count+=1
240
241     # Loop that generates the specificed number of bins
242     for a in range (1, slotn):
243         bpy.ops.mesh.primitive_cube_add(size=1, location=(-n*lgapx + (w/
    slotn)*a -extra/2,
244             y, z - n*gapz - ((z - n*gapz)*0.5)))
245         bpy.ops.transform.resize(value=(cube*0.15, bret, z - n*gapz),
246             constraint_axis=(True, True, True))
247         bpy.ops.rigidbody.object_add()
248         bpy.context.object.rigid_body.type = 'PASSIVE'
249         bpy.context.object.rigid_body.use_margin = True
```

```
250         bpy.context.object.rigid_body.collision_margin = 0.0001
251         bpy.context.object.rigid_body.friction = 0
252         obj = bpy.context.object
253         bin_collection.objects.link(obj)
254 
255 
256 
257 
258     # Create a new collection for physics objects
259     sphere_collection = bpy.data.collections.new('spheres')
260     bpy.context.scene.collection.children.link(sphere_collection)
261     for b in range (1, cubex):  # Loops that generate the number of
    particles specificed along a single row only
262             bpy.ops.mesh.primitive_ico_sphere_add(radius=cube*0.5,
263                 location=((-1.5*cube-ramp*0.5*0.71)+ (2*b*(1.5*cube+ramp
    *0.5*0.71))/cubex,
264                 y, z + 2*gapz + ramp*0.5*0.71 +1.25*cube))
265             bpy.ops.rigidbody.object_add()
266             bpy.context.object.rigid_body.type = 'ACTIVE'
267             bpy.context.object.rigid_body.use_margin = True
268             bpy.context.object.rigid_body.collision_margin = 0.0001
269             bpy.context.object.rigid_body.friction = 0.0
270             bpy.context.object.rigid_body.restitution = 0.25
271             bpy.context.object.rigid_body.linear_damping = 0.6
272             bpy.context.object.rigid_body.angular_damping = 0.3
273             bpy.context.object.rigid_body.mass = cubemass
274             obj = bpy.context.object
275             sphere_collection.objects.link(obj)
276 
277     # Duplicate physics objects in z direction
278     bpy.ops.object.select_same_collection(collection="spheres")
279     for n in range(0,cubez):
280         bpy.ops.object.duplicate_move(OBJECT_OT_duplicate={"linked":
    False, "mode":'TRANSLATION'},
281         TRANSFORM_OT_translate={"value":(0, 0, 0.5), "orient_type":'
    GLOBAL',
282         "orient_matrix":((1, 0, 0), (0, 1, 0), (0, 0, 1)), "
    orient_matrix_type":'GLOBAL'})
283 
284     text='baking f='+str(f)+'Hz'+'
    ----------------------------------------'
285     print(text)
286 
287     # BAKE ALL DYNAMICS
288     # Set Start and End of Simulation Bake
289     bpy.context.scene.rigidbody_world.point_cache.frame_start = 0
290     bpy.context.scene.rigidbody_world.point_cache.frame_end = endframe
```

```
291
292     bpy.context.scene.rigidbody_world.substeps_per_frame = sspf # set
    steps per second
293     bpy.ops.ptcache.bake_all(bake=True)
294
295     # SAVE AND EXPORT DISTRIBUTION
296
297     # Go to the last frame to save the ball locations
298     bpy.context.scene.frame_set(endframe)
299
300     # Create filename
301     name='f'+str(f)
302     filename="%s.csv" % name
303
304     # Save ball locations in .csv
305     bpy.ops.object.select_same_collection(collection="spheres")
306     bpy.ops.object.visual_transform_apply() # Applies visual
    transformation
307     os.chdir('C:\\Users\\Aidan\\OneDrive\\BiasedGaltonData\\3mmsim') #
    File directory
308     with open(filename, 'w') as csvfile: # File name
309         writer = csv.writer(csvfile, delimiter=',')
310         for sphere in bpy.data.collections['spheres'].objects: #Loops
    through collection of Spheres
311             writer.writerow([sphere.location[0],sphere.location[2]]) #
    Records x and z position
312             sphere.location[2]
313
314     print('write complete')
315
316     text=str(lastf-1-f)+' remaining'
317     print(text)
318
319
320 print('ALL COMPLETE **********************************************')
```

The following script was used to process the frames from the slow-motion video of the mechanical oscillating Galton board. It isolates the balls and calculates the entropy from each processed frame.

```
% Author: Aidan Lee
% Last Updated: 09/04/2021
%
% This script calculates the entropies in each frame of the slow-motion
% video of the mechanical oscillating Galton board experiment

% clear existing variables and values in the workspace
clear

% define directory
cd('D:\BlenderSaves\OGB\120fpsexperiment\10hzstabilized');

% read all files with .png extension
files = dir('*.png');



%% Get mode pixel values from first 185 frames
% define the sampling interval
firstframe=1;
lastframe=186;

for n=firstframe:lastframe

    % read and store the pixel values from png file
    img = imread(files(n).name);
    data185(:,:,n)=img(940:1650,1125:2550,1);
    n=n+1;

end

% calculate and store the mode of each pixel
bk = mode(data185,3);

%% Process the images to calculate the entropy
for n=1:length(files)

    % read, store and crop the image
    img = imread(files(n).name);
    data(:,:)=img(940:1650,1125:2550,1);

    % remove the mode pixel values from the image
    imagesc(double(data(:,:))-double(bk));
```

```
44
45      % apply threshold filter to isolate the dark balls
46      bw=double(data(:,:))-double(bk)<-30;
47
48      imagesc(bw);
49      E(n) = entropy(bw); % calculate the entropy from the image
50      colormap gray
51      pause(0.1)
52
53      % clear the stored image to free memory for the next image
54      clear img
55      clear data
56  end
57
58  % export the calculated entropies to a .csv file
59  writematrix(E,'mech0GB_entropies.csv')
```

The following script computes the intersections of a phase-space trajectory with a defined cutting plane. The phase-space coordinates are computed separately from this script.

```
% Author: Aidan Lee
% Last Updated: 27/03/2021
%
% This script calculates the intersections to generate a Poincare map
% using the cutting-plane method.
%
% Separate scripts are used to read and store data.

close all

%% Parameters of figure (colour, font size, etc.)

% Define colours (each dynamic system is colour-coded)
simpen=[158 57 135]./255; % simple pendulum
blue=[0 113 188]./255;    % magnetic pendulum conservative
orange=[235 92 21]./255;  % magnetic pendulum dissipative
dubpen=[30 107 138]./255; % double pendulum
spcb=[240 161 21]./255;   % chaotic spaceballs
pspcb=[19 96 102]./255;   % periodic spaceballs
rc=[162 76 76]./255;      % rolercoaster chaos
color=pspcb;              % select colour

fsize=48*2; % font size of axis labels
tsize=24*2; % tick font size
msize=15; % marker size

%% Parameters of phase space and Poincare map

% Define state variables that form the 3D phase space trajectory
xdot=velocities0(:,1);
ydot=velocities0(:,2);
zdot=velocities0(:,3);

% Define the normal vector of the cutting-plane
n=[0 0 1];

% Define a point in the cutting-plane
V0=[0 0 0];

coordinates=[xdot ydot zdot]; % points of the phase space trajectory
nvals=length(coordinates)-1;  % number of values that form the
   trajectory
intersections1=zeros(1,3);    % storage of intersection points
p=1;
```

```
44
45 %% Computation of Poincare Map using cutting-plane method
46 for frame=1:nvals
47
48     % point and subsequent point that forms line segment of trajectory
49     P0=coordinates(frame,:);
50     P1=coordinates(frame+1,:);
51
52     % check if this line segment intersects the cutting-plane
53     [I,check]=plane_line_intersect(n,V0,P0,P1);
54
55     % if the line segment intersects the cutting-plane, check if the
       angle
56     % between the line segment vector and the plane's normal vector is
57     % acute. This is to sample only 1 side of the plane.
58     if check==1
59
60         u=P1-P0;
61         dotprod=dot(u,n);
62         magprod=norm(u)*norm(n);
63
64         angle=acos(dotprod/magprod);
65         angdeg=rad2deg(angle);
66
67         % if the angle is acute, store the intersection point
68         if angdeg<90
69             intersections1(p,:)=I;
70             p=p+1;
71         end
72     end
73 end
74
75 % plot the intersections
76 plot(intersections1(:,1),intersections1(:,2),'.','markersize', ...
77     msize, 'color', color); axis equal
78
79 a = get(gca,'XTickLabel');
80 set(gca,'XTickLabel',a,'fontsize',tsize);
81 set(gca,'LineWidth',4,'TickLength',[0.025 0.025]);
82
83 xlabel('$\dot{x}$ (m/s)', 'Interpreter','latex', 'FontSize', fsize);
84 ylabel('$\dot{y}$ (m/s)', 'Interpreter','latex', 'FontSize', fsize);
85
86 set(gcf, 'Renderer', 'painters', 'Position', [-200 -200 2400 1600])
87 set(findall(gca, 'Type', 'Line'),'LineWidth',12);
88
89 axis equal
```