BEng Mechanical Engineering Group Project

Controlled Electromagnetic Levitation in Multiple Dimensions

November 29th 2018

Group WR1

Maurice Rahme
Yidan Xue
Max Benson
Emil Hansen
Daniel Carbonell

Contents

1 Introduction

Magnetic levitation is a rapidly developing field of engineering, with vast engineering application found across many industries such as transportation, high precision motors and energy storage. The benefits of the technology arise from its no contact operations leading to very low losses and minimal maintenance (Schweitzer and Maslen, 2009). Additionally, magnetic bearings can be used in vacuums where conventional bearings can not be used as the lubricant would evaporate.

Actively controlled levitation is a crucial component in high speed transportation. Tolerances in the wheels and the track no longer become an issue in these applications; it is currently used in the SCMaglev railway system in Japan and is an essential component to the rapidly developing field of Hyperloop technology. Audio company Bose also displayed a prototype car using magnetic suspension which used actively controlled magnetic levitation to stabilise the vehicle and was shown to be far superior to industry standard shocks.

Furthermore, the low friction properties are being exploited for energy storage by companies such as NASA and Teraloop where high speed flywheel technology is being used to balance demand from the power grid. They apply energy to the flywheel when demand is low and then the inertial energy is converted back to electricity when demand is high - working as an exceptional companion to solar and wind energy.

This paper focuses on the stable levitation of a steel ball to explore the principles of magnetic levitation and the control theory that underlies the technology. After a model was developed and a test rig was made to balance the ball in place at a set height, a more complex two dimensional model was then made to actively control the height of the ball as well as its longitudinal travel.

Tuning such systems is not simple, thus a machine learning approach was adopted to find suitable PID values for stable operation. A genetic algorithm was used which produced increasingly suitable PID values as it was left to run over time. The algorithm provided a quicker, easier and more accurate solution compared to manual tuning.

2 Literature Review

2.1 Theoretical Implementations of Magnetic Levitation of Ball Bearings

There is an abundance of papers suggesting methods of deriving transfer functions for magnetic levitation of spheres. However, they often do not solve the nonlinear and unstable properties of the system perfectly. Indeed, finding past research material on feasible control of steel ball levitation proved difficult. For example, Sorour (2015) introduced the method of deriving the transfer functions and linearising the system using Taylor Series. He also designed a lead-lag compensator and tuned the system via the root locus method. However, there were some errors in their derivation and implementation. At the same time, Hlebowitsh focused on the development of control methods for 2D and 3D systems. He suggested several innovative ideas for linearising the system, but some of his assumptions might not be reasonable and his control system does not satisfy the design requirements. In addition, some research (Kumar and Minz, 2016; Gazdos, Dostal and Marholt, 2011) showed good results, but they were brief in derivation process and tuning method. At the same time, some research (Hajjaji and Ouladsine, 2001) achieved acceptable results in a detailed way, but it was too difficult to repeat such a complex process in a short period of time. In summary, after reviewing previous research outputs, complete and proper theory for this project was not found which meant a pioneering theory to solve this problem needed to be developed by ourselves. The literature review also suggested that traditional tuning methods such as Ziegler-Nichols would be ineffective for a system of this non-linearity and instability. Duriez et al. (2017) and Amaral et al. (2005) suggested that genetic algorithms would be the optimal way of tuning PID controllers under these conditions which served as a starting point for exploration.

2.2 Practical Implementation of the System

Technical specifications of such a system proved difficult to find, partially because the majority of academic papers focused on the transfer function and theoretical behaviour of the system and because the papers that did talk about the system used equipment far outside of the price ranges available to us. Therefore the literature review focuses heavily on exploring datasheets for various sensors (explored in sections 8.3.1 through 8.3.4)

3 Project Management

At the start of the project, there was apparent value in assigning a project manager to the group because of the scope of the project. The project manager was responsible for creating a detailed project plan, taking minutes at meetings, distributing the roles and tasks to the group members, and ensuring that the group stayed on track to complete the project on time.

The project plan was composed of a Gantt chart which detailed every major task to be completed and was presented in a comprehensible visual format. It is clear from the chart that there were many important tasks to complete and the project manager put an emphasis on meeting strict time management deadlines. This was deemed to be of great importance in a project with such challenging goals but limited by the amount of time with which to achieve them.

The major task areas were assigned based on previous experience of the members of the group and interest of individuals to work on a specific area of interest to them. These major areas were magnetic simulation and 1D/2D transfer function derivation (Yidan Xue), 1D/2D control system design and VR simulation of the system (Maurice Rahme), designing and building the physical system (Daniel Carbonell), sensor and control research and implementation (Emil Hansen), software and controller design, implementation and tuning (Emil Hansen and Max Benson).

After the interim presentation and having discussed the feedback given, the project manager decided with the group to re-evaluate the goals and the feasibility of achieving them within the tight budget and timescale. Having discussed this it was decided to continue working towards the initial goals. However, the goals for 2D simulation were adjusted. Instead of controlling the steel ball between any two points - as has never been achieved before using only an electromagnet - it was decided to approximate this to small movements from a target point. However, the long movement and 3D movement could still be achieved by this way, if several electromagnets were laid out in a grid (Hlebowitsh, 2012).

4 Magnetic Simulation

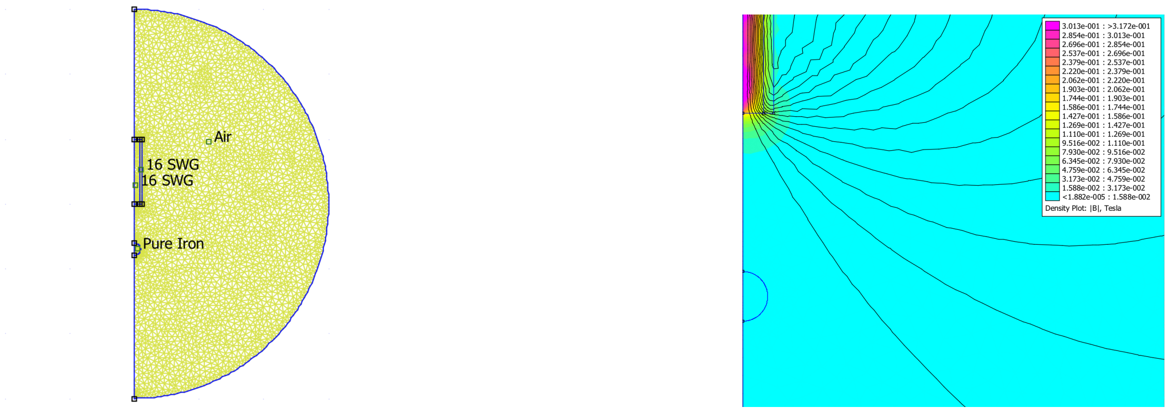
Due to the complexity of the physical system, a magnetic simulation was implemented to find an accurate relationship between the magnetic force on the steel ball and its displacement. Finite Element Method (FEM) is often used when solving magnetic problems. It divides the domain into a large number of small sub-domains, repeatedly solving Maxwell's Equations inside each small block and achieves solutions for the whole domain. Currently, there are several magnetic simulation software packages using this technique and they can be used for the problem laid out above.

4.1 Software Selection and Simulation Process

Finite Element Method Magnetics (FEMM), ANSYS AIM and ANSYS Maxwell were compared before beginning simulations. Maxwell is one of the most used professional magnetic simulation software packages developed by ANSYS (2018a). However, it was not used because of its complexity and steep learning curve. Hence simulations were completed using FEMM and ANSYS AIM to choose the better one for this simulation.

FEMM (2014) is a 2D magnetic simulation software, but it can solve 3D axisymmetric cylindrical coordinate problems. The solenoid and ball were modelled in an axisymmetric coordinate system and meshed in the the sphere domain as Figure 1 (a) shows. After solving the magnetic field inside the domain shown in Figure 1 (b), the force on the steel ball could be integrated. Then, the ball was moved along the y axis. At different positions, the current inside the coil could be adjusted by changing the current flux density (4.1) to find the current which could balance the ball. This was done as FEMM does not natively support alterations in current.

$$J = \frac{4I}{\pi D^2} \tag{4.1}$$



(a) Mesh (b) Solution

Figure 1: 2D Magnetic Simulation by FEMM

However, a huge difference was found between the FEMM results and the experimental data. The coarse mesh generated by FEMM was one possible reason for this error, especially the mesh generated inside the ball. From the calculation procedure document, it was found that only 5004 nodes and 9629 elements were created. It didn't seem enough to be able to describe the complicated magnetic field. Another reason might be the integration process from 2D Cartesian coordinates to 3D cylindrical coordinates could not reflect the 3D physical system accurately. Hence this method was not chosen in the end.

ANSYS AIM (2018) is another 3D magnetic simulation software developed by ANSYS and uses a similar solver to Maxwell. It can solve static magnetic field problems in 3D and is free for academic use. After getting the 19.2 Academic version from ANSYS, the 3D geometry of solenoid and steel ball was created based on the physical design and generated a computing domain around them. The geometry then was transported to the physical solver. When setting up the solver, proper materials were assigned to parts and the current was generated inside the coils. Then, a finer mesh option and more accurate solution option were selected. Finally, the solution could be calculated as Figure 2 shows. This process was repeated by changing the current until the magnetic force on the steel ball could be balanced by its gravitational force.

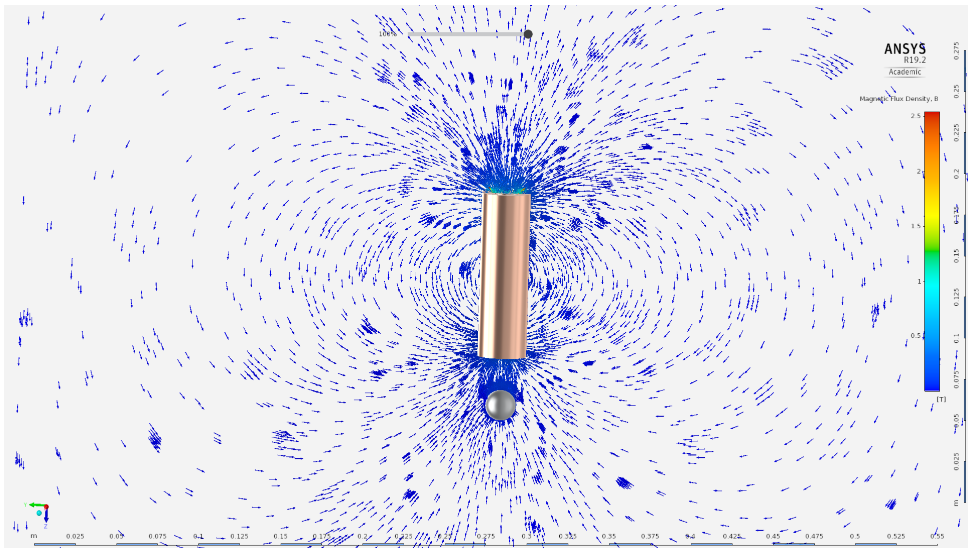


Figure 2: 3D Simulation Results in ANSYS AIM

Next, the position of the steel ball was changed in the geometry and the process was repeated to obtain the current required to balance the ball at different positions along the vertical axis. The results were listed in Table 1 and plotted in MATLAB as shown in Figure 3. These solutions matched with the experimental data and were used for deriving the transfer function.

Table 1: 3D simulation results data

Displacement(m)	Current(A)
0.05	46
0.045	37
0.04	28
0.035	20.5
0.03	14.5
0.025	9.5
0.02	6.0
0.015	3.3
0.01	1.05

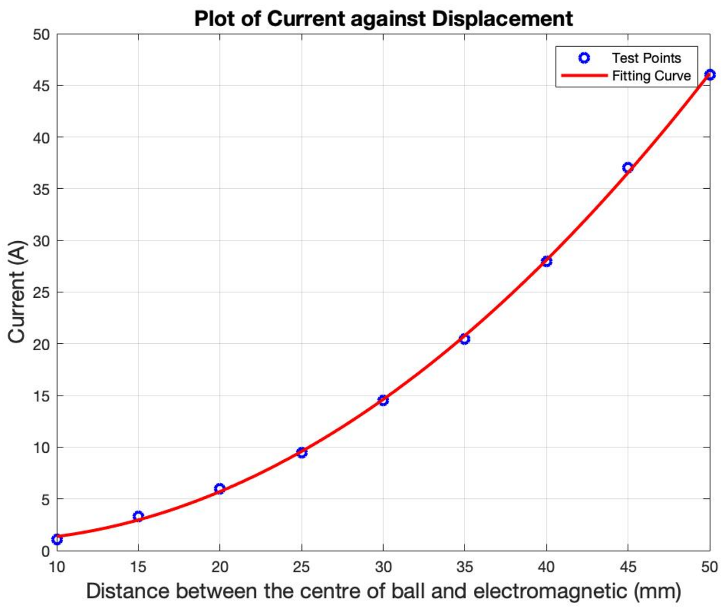


Figure 3: Current required to balance the gravity force at different positions

4.2 Simulation Results Analysis

A quadratic curve was fitted to the simulated testing points as shown in Figure 3, suggesting a relationship between current and displacement described by equation 4.2.

$$I = k(x + x_0)^2 \tag{4.2}$$

Where k is a constant based on the physical system and x_0 is the offset. Hajjaji and Ouladsine (2001) achieved similar experimental results that show the square of current was proportional to the displacement in the region near the solenoid. At the same time, there are lots of papers assuming the current is proportional to the displacement when deriving their transfer functions (Gazdos, Dostal and Marholt, 2011; Kumar E and Jerome, 2013). However, those assumptions do not seem to be reasonable due to the nonlinear properties of the magnetic field (Woodson and Melcher, 1968). In addition, the steel ball cannot be simplified as a point in this practical problem. This will cause a variation of the magnetic flux through the ball as its position changes. Thus the simulation data seem more suitable for the purposes of this project.

5 Derivation of the 1D Transfer Function

5.1 Physical Model and Assumptions

There are several methods of obtaining the transfer function for this system (Barie and Chiasson, 1996; Gazdos et al., 2011; Kumar E and Jerome, 2013). All methods follow a similar procedure of building the mathematical model from a free body diagram, linearising the system at the operating point, calculating the original unstable transfer function and then stabilising the transfer function.

To derive the 1D transfer function, a similar process was followed. Firstly, the copper coils are assumed to be perfectly wound and the materials of the ball and the core are uniform. Secondly, there are only gravitational and magnetic forces affecting the movement of the ball which meant the effect of air resistance could be neglected. Thirdly, the effect of magnetic hysteresis was neglected. Hence the magnetic force was a formula of the current and it did not change with time for a given current. In addition, the relationship of current and displacement was linear at the operating point and was accurate over a small area. However, this did not mean the relationship was linear for the whole region, the relationship was given in the previous section.

Based on these assumptions, the magnetic levitation system was modelled as shown in Figure 4.

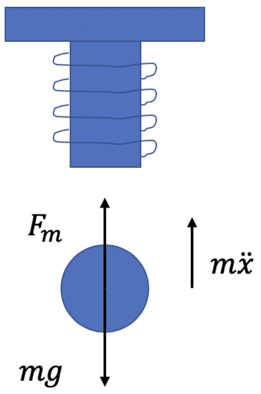


Figure 4: Physical model of the steel ball and solenoid

5.2 Calculation Process

Due to Newton's second law, the movement of the steel ball could be expressed as:

$$m\ddot{x} = F_m - mg \tag{5.1}$$

Where m is the mass of the steel ball which is $0.028\,kg$, x is the position of the ball and F_m is the magnetic force.

Then, a damping term could be included and the magnetic force could be calculated as:

$$m\ddot{x} - k_d\dot{x} = \frac{k_p {k_i}^2 k_c u^2}{x^2} - mg \tag{5.2}$$

Where k_d is the damping constant which is assumed as 0.02 Ns/m, k_p is the position sensor gain which is calculated as $68.18\,V/m$ based on our sensor design, k_i is the amplifier gain which is $0.29\,A/V$ and where k_c is the coil constant which is about $1 \times 10^{-6}\,Nm^2/A^2$ (Gazdos et al., 2011). It is noted that the simulation results are linearised at the equilibrium position of $x = 3cm$ to get a transfer function suitable for a Laplace Transform.

Thus a matrix equation (Barie and Chiasson, 1996) could be written for this linearised model as

$$x = Ax + Bu \tag{5.3}$$

$$x = \begin{bmatrix} x & \dot{x} \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 \\ \frac{2g}{k_i u \sqrt{\frac{k_c}{mg}}} & \frac{k_d}{m} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{2k_p g}{u} \end{bmatrix} \tag{5.4}$$

Next, the transfer function could be calculated (Gazdos et al., 2011) as

$$G(s) = [1\ \ 0] \cdot (s \cdot I - A)^{-1} \cdot B \tag{5.5}$$

where I is the identity matrix. Hence the transfer function of the system is

$$\frac{U(s)}{X(s)} = G(s) = \frac{767}{s^2 - 0.7s - 1723} \tag{5.6}$$

However, this transfer function is not stable, as there is one pole ($s = 41.86$) on the right half of the complex plane. In order to stabilise the transfer function, spectral factorisation method is implemented. It can keep the amplitude information of the transfer function while making it stable (Overdijk, van de Wouw and de Kraker, 2001). Implementing this technique in the transfer function, the denominator of the new transfer function $f(s)$ could be obtained by

$$f(s)f(-s) = g(s)g(-s) \tag{5.7}$$

which is an equation on complex plane and $g(s)$ represents the denominator of $G(s)$.

Solving this complex equation gives

$$F(s) = \frac{b_0}{s^2 + a_1 s + a_0} \tag{5.8}$$

$$b_0 = \frac{2k_p g}{u} \quad a_1 = -\frac{k_d}{m} \quad a_0 = -\frac{2g}{k_i u \sqrt{\frac{k_c}{mg}}}$$

After replacing the coefficients with the variables we have mentioned before, we could get the stable transfer function of

$$\frac{U(s)}{X(s)} = F(s) = \frac{64}{s^2 + 83s + 1723} \tag{5.9}$$

which would be used in the control system.

6 Control System Design

6.1 Genetic Algorithm for PID Parameters

6.1.1 Genetic Algorithm

A genetic algorithm (GA) was used for two separate elements of the project.

To tune the “1D” controller, GA was employed because, given the nonlinear nature of the system, the spectrum of appropriate Proportional, Integral and Derivative (PID) controller values was greatly reduced. Furthermore, when trialled, the Ziegler-Nichols method did not produce stable results suitable for further hand-tuning. Secondly, for our physical 1D system, similar issues were found with the Ziegler-Nichols approach, so a GA was designed and written in Python to automatically find suitable PID values to be used by the control system (Amaral et al., 2005).

The genetic algorithm as a concept is founded on Darwinism and natural selection (survival of the fittest). It works on the basis that individuals in a population have different traits and these traits are heritable. If an individual carries a trait that is better suited to the environment it is in, it is more likely to survive than an individual with less beneficial traits. Those individuals that survive can then repopulate, carrying their beneficial traits forward to the succeeding population. However, Darwinism also states that mutations can occur, and these play an important role in developing the individuals beyond the traits of their initial population. While some mutations may have negative consequences, other mutations cause new beneficial changes which might result in an even better individual than the previous best.

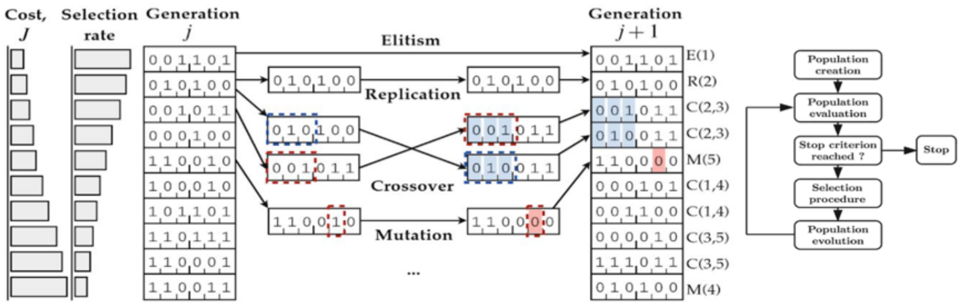


Figure 5: A high-level map of the Generic Algorithm showing Elitism, Replication, Crossover and Mutation operations. The algorithm’s overall procedure is also shown in the State Flow model on the right-hand-side of the figure.

As the diagram above (figure 5) shows, the cost ‘J’ is evaluated for each individual in the population of generation ‘j’; the selection rate is inversely proportional to the cost produced by each individual. To constitute generation ‘j+1’, four operations can be done: the first is elitism, where the best individual of each generation is immediately passed to the next. This is important because it allows the algorithm to avoid the loss of good solutions due to genetic drift through over-mutation. Next is crossover, where two individuals in the population can exchange some properties and retain others as shown in the figure. Additionally, replication can be done, whereby a random individual is also passed to the next generation without crossover of properties. Finally, mutation can be done whereby a random parameter of the individual is changed to a random value. This allows for a more exhaustive exploration of possible PID values.

6.1.2 1D Simulation

For the theoretical control system, the GA was implemented using MATLAB’s GA function. Here, the transfer function is set up by defining *s* as a Laplace variable. The options parameter within the GA function allow for outputting simulation data, which provides greater insight into the evolution of the algorithm when examined; they also set the population and generation sizes. The cost function is present within the ‘pidtest’ code, which is used to draw the resultant transient response for given PID values, and is computed for each generation of the GA’s execution; the cost function returns the square of the settling error plus the square of the settling time.

The GA function’s notation in MATLAB is as follows: ‘ga(@(K)pidtest(G,dt,K)’, where the pidtest function contains the matrix of PID parameters (K), and the @ command focuses K as the parameter to be changed when attempting for a lower cost function, ’J’. ’G’ is the system’s transfer function, derived in the previous section, and ’dt’ is the time step, set to 1ms.

Using the data saved in the GA function’s execution, it is possible to view the evolution of the K matrix, which holds the PID values shown in Figure 6:

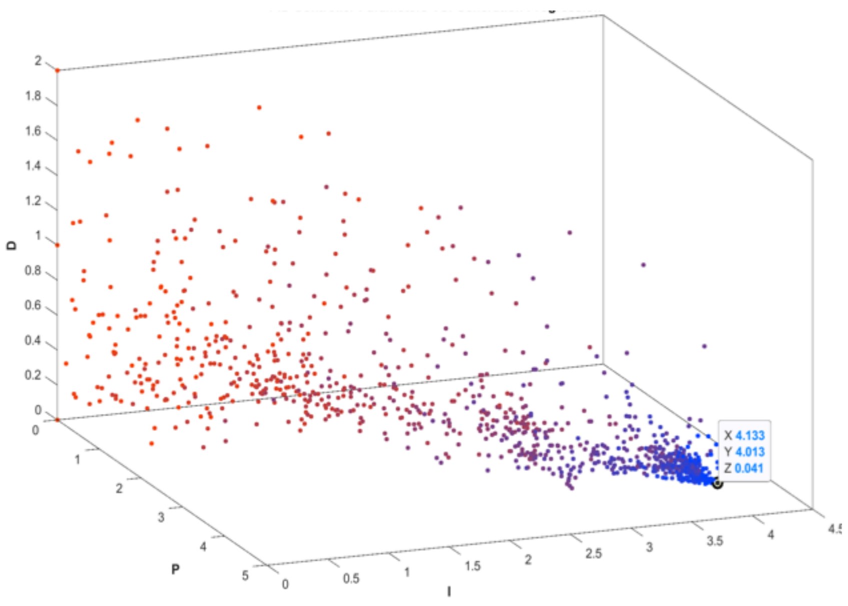


Figure 6: The evolution of the P, I, and D parameters (denoted by X,Y,Z respectively) is shown for the execution of the GA. The reduction in mutation over subsequent generations is evident here, whereby the data points are more concentrated for later generations, and more scattered (highlighting the exploratory nature of the mutation operation) in the beginning. Earlier generations tend towards a lighter red colour, while later generations tend toward a lighter blue colour.

Furthermore, the evolution of cost is also shown for each population with respect to generation progression, as shown in Figure 7:

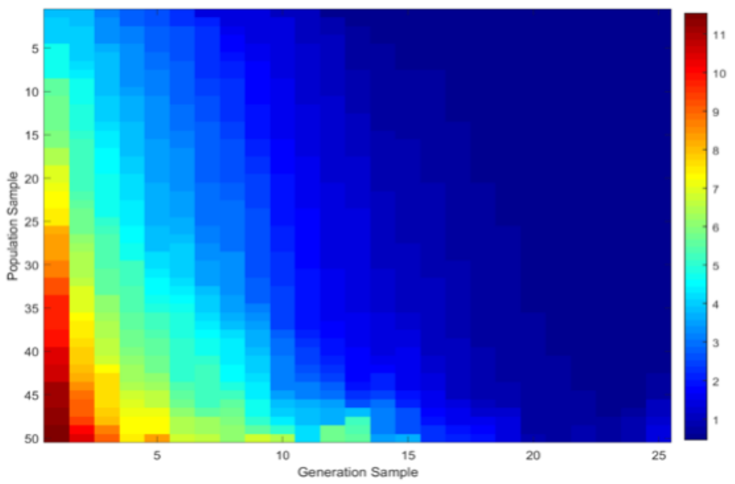


Figure 7: This chart shows the progression of cost versus the GA's evolution. The colour bar on the right assigns resultant cost values to the 'jet' colour gradient. The highest cost achieved was 11.913. Note that the individuals in each population from 1 to 50 are organised by decreasing selection rate to reflect the fundamental workings of the GA.

Notice the steady decrease in mutations per generation denoted by high costs for later generation samples. The lowest cost obtained was 0.4633.

Figure 8 shows the best transient response plotted for the nonlinear system, with no overshoot and a 5ms settling time.

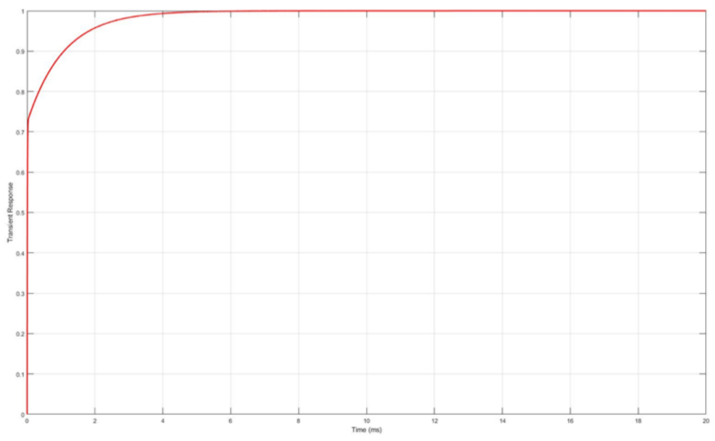


Figure 8: The system's transient response for the best PID parameters found by the GA.

6.1.3 1D Physical System

The implementation of the genetic algorithm was similar for the physical system and was designed as follows. An initial population of 50 "chromosomes" with constrained random P, I and D parameters is created. The program iterates over all the chromosomes in the population, measuring the "fitness" of each based on a predefined fitness function. Here, fitness is merely the inverse of cost and is therefore maximised. This fitness function computes the inverse of the square of the overall error of the system for a fixed number of samples. The chromosomes rated with highest fitness are given a larger probability of being picked for crossover, thus improving the overall population fitness over time. The mutation rate determines whether a new child chromosome will mutate. This mutation rate was 0.33, a value obtained from a paper that used a genetic algorithm to tune the PID parameters for a magnetic levitation rig similar to ours (Pedersen and Yang, 2006).

6.2 Simulink Model and Virtual Reality Demonstration

To provide an appropriate PID controller for the real system, it was essential to hand-tune the preliminary values gathered by the GA in the context of a nonlinear system. For this endeavour, the nonlinear plant was modelled in SimuLink.

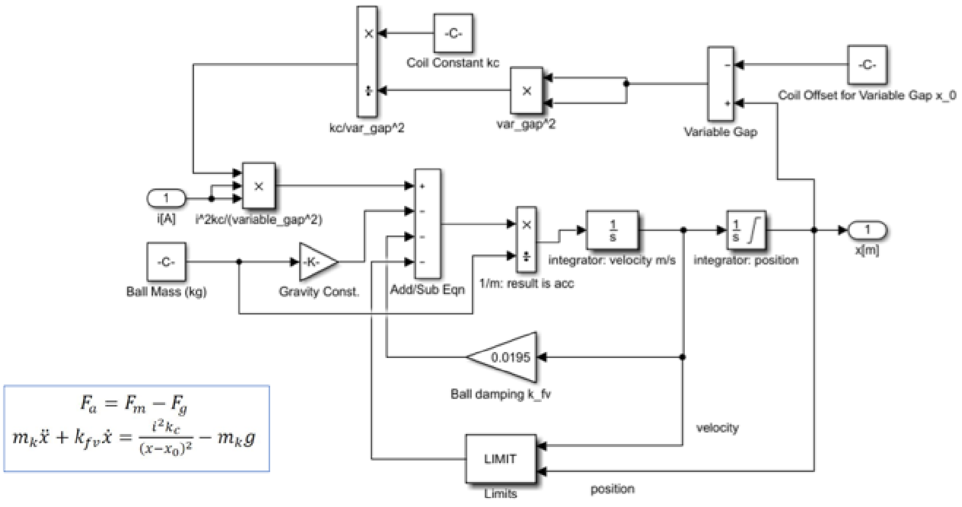


Figure 9: Nonlinear Plant Model represented using Simulink Blocks.

This plant was subsumed into a Plant Model block, and a PID controller block was added in cascade, forming the feedback control loop as shown below in Figure 10.

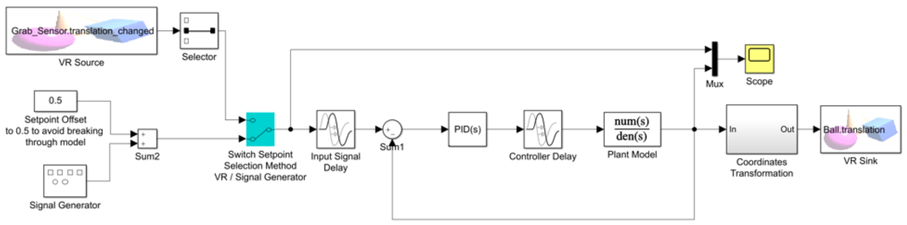

Grab_Sensor.translation_changed
VR Source
Selector
0.5
Setpoint Offset to 0.5 to avoid breaking through model
Sum2
Signal Generator
Switch Setpoint Selection Method VR / Signal Generator
Input Signal Delay
Sum1
PID(s)
Controller Delay
num(s)/den(s)
Plant Model
Mux
Scope
In
Out
Coordinates Transformation
Ball.translation
VR Sink

Figure 10: Full Simulink Model including two selection methods (Signal Generator and VR 'Grab Sensor') and an oscilloscope to evaluate the behaviour of the controlled response.

The Simulink model includes a signal generator which acts as the controller input. It also implements a virtual oscilloscope for a visual comparison of the control input and measured output.

Notably, the Virtual Reality (VR) 'Source' and 'Sink' blocks tie the system to a VR interface which allows for visual interpretation and demonstration. This was built within MATLAB's VR World environment using the provided tutorials and examples whereby an STL model of the real system was imported as shown in Figure 11. Furthermore, a Grab sensor was included so that a set point for the ball can be set by interacting with the VR environment using the computer mouse; this functionality is allowed using the 'VR Source' block, while the 'VR Sink' block provides the visual representation.

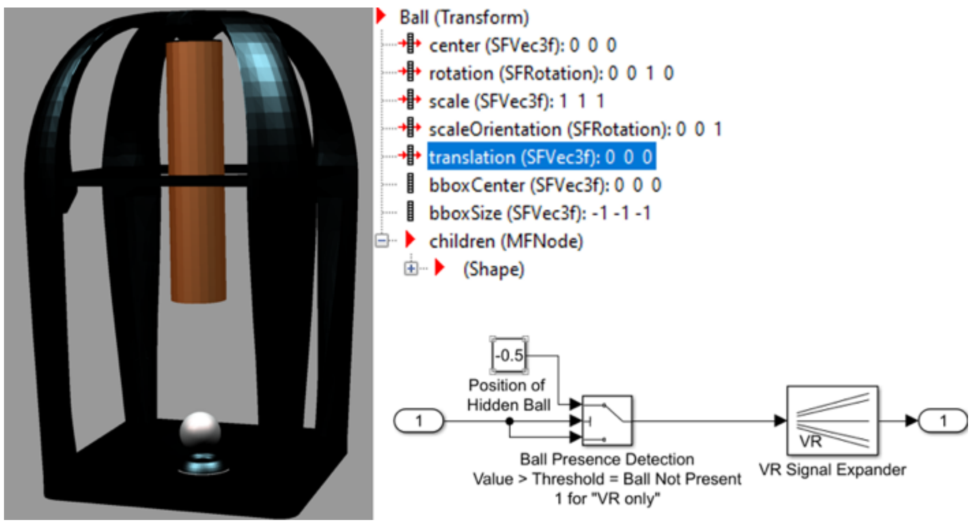

Ball (Transform)
center (SFVec3f): 0 0 0
rotation (SFRotation): 0 0 1 0
scale (SFVec3f): 1 1 1
scaleOrientation (SFRotation): 0 0 1
translation (SFVec3f): 0 0 0
bboxCenter (SFVec3f): 0 0 0
bboxSize (SFVec3f): -1 -1 -1
children (MFNode)
(Shape)

Figure 11: The VR environment used for demonstrative purposes is shown on the left-hand-side of the figure. In the top-right, the Ball (Transform) node is shown, which moves the ball in the VR space via the translation branch using the output from the Simulink model as processed through the 'Coordinate Transformation' block, shown in the bottom right.

The crucial part of the VR environment is the 'Ball' transform which allows the virtual ball to move. The 'translation' branch of the 'Ball (Transform)' node takes inputs from the 'Coordinate Transformation Block' shown in in both figures 10 and 11, which itself takes input from the ball's position as reported by the Simulink model. The 'Coordinate Transformation Block' also hides the ball if it contacts the 'magnet' or 'floor' to avoid breaking the VR model.

6.3 Control System Tuning and Stability Analysis

Upon running the simulation using the nonlinear plant model controlled by PID values for a linearized system, a significant 25% overshoot was observed. This discrepancy between the linearized and nonlinear models was expected, and is shown in Figure 12 below:

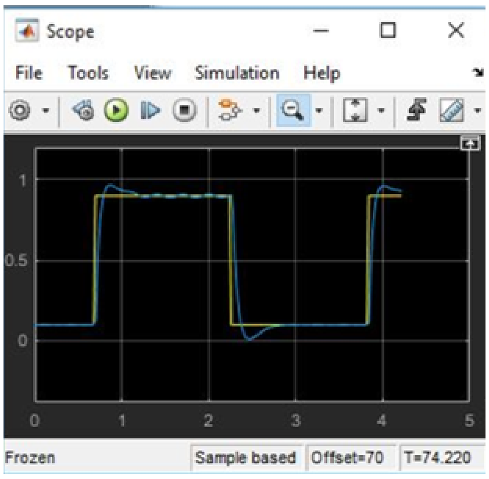
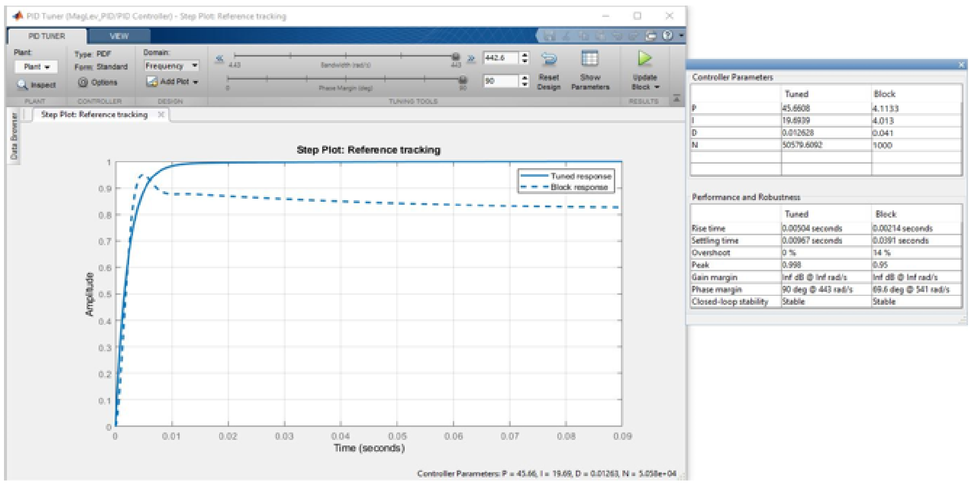


Figure 12: The output from the oscilloscope in the Simulink model compares the control signal (yellow) to the output position signal (blue). Here, the sensor voltage represents the ball's vertical position. The maxima and minima set for the controller are 0.8 and 0.2 respectively.

To improve the response, the PID controller was tuned with respect to the nonlinear plant model in Simulink; the initial and final hand-tuned responses are shown alongside their respective PID values in Figure 13. The discrepancy in response between the tuner environment and the simulation is due to the parameter 'N', the filter coefficient which tends closer to a true derivative (D) controller value when increased. It was set low here to avoid crashing the simulation and did not represent the actual Derivative value for the controller. However, the hand-tuned system provided its own value which was usable without issues.



Controller Parameters: P = 45.66, I = 19.69, D = 0.01263, N = 5.058e+04

Controller Parameters

	Tuned	Block
P	45.6608	4.1133
I	19.6939	4.013
D	0.012628	0.041
N	50579.6092	1000

Performance and Robustness

	Tuned	Block
Rise time	0.00504 seconds	0.00214 seconds
Settling time	0.00967 seconds	0.0391 seconds
Overshoot	0 %	14 %
Peak	0.998	0.95
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	90 deg @ 443 rad/s	69.6 deg @ 541 rad/s
Closed-loop stability	Stable	Stable

Figure 13: Simulink's PID hand-tuner. The result was a stable system with 0% overshoot, a 9ms settling time, and infinite gain margin and a 90 degree phase margin. This is an improvement from the previous settling time of 39ms for the nonlinear model.

To show proof of stability, the simplest method of analysis was to observe the Bode Plot. First, consider the untuned system, shown in Figure 14:

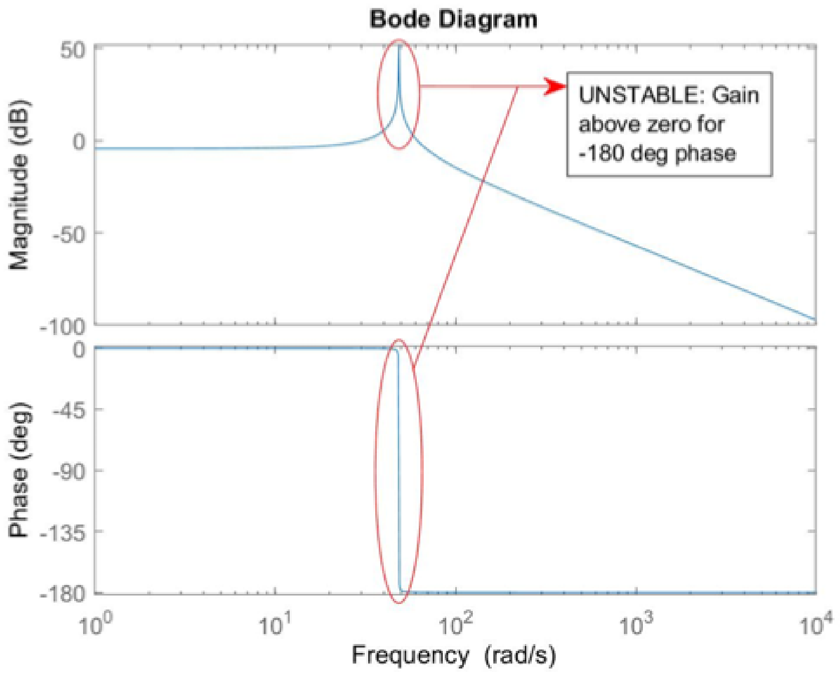


Figure 14: Bode plot for the untuned system showing instability at 31 rad/s due to system resonance at this frequency.

Due to the asymptote in the bode plot, arising from the system's natural frequency, the system gain rises above zero for a phase of -180 degrees, indicating instability.

Comparatively, the tuned system has the following Bode response, shown in Figure 15:

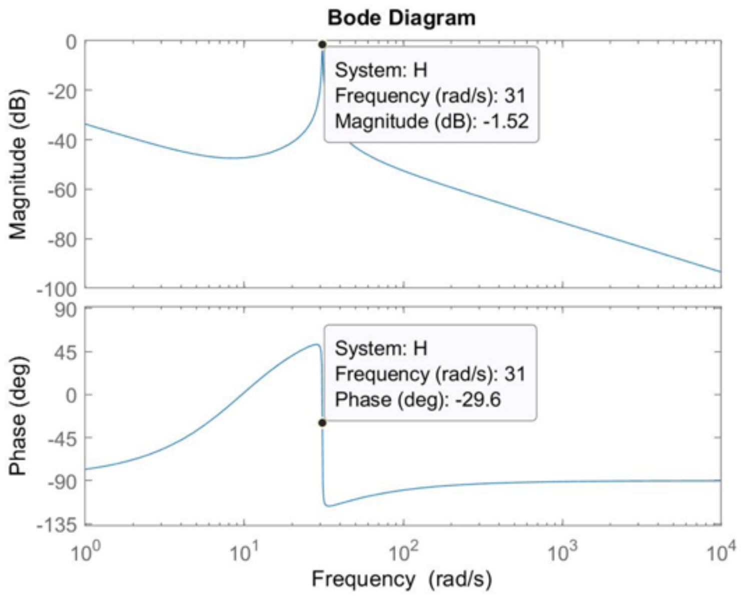


Figure 15: Bode plot for the hand-tuned system showing stability as the gain is always below zero and there is a 90 degree phase margin.

Notice the 90 degree phase margin, and that the gain never moves above zero, despite system resonance at 31Hz. The gain continues to decrease exponentially past 31 rad/s. This matches with the observed simulation behavior as high frequency square-wave inputs provide control akin to PWM operations whereby the result is a static set point for the ball taken from the observed average voltage of the control signal.

Notably, the Nyquist plot shown in Figure 16 confirms stability since there are no encirclements about the $(-1 + 0j)$ point denoted with a red cross. It also corroborates the presence of system resonance at 31.4 rad/s, since the plot's encirclements enter the real negative axis at this frequency which denotes a non-infinite phase margin; however, it is still stable.

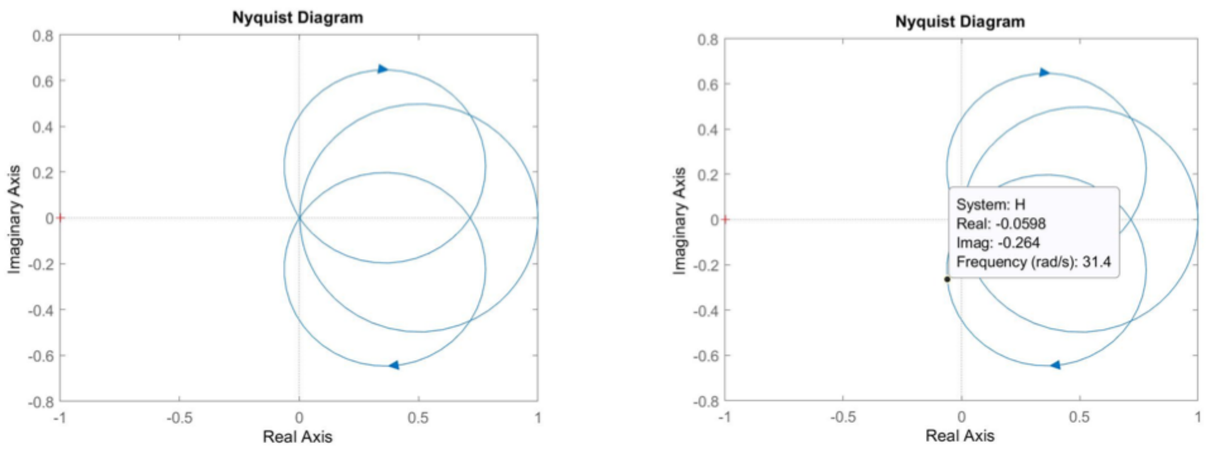
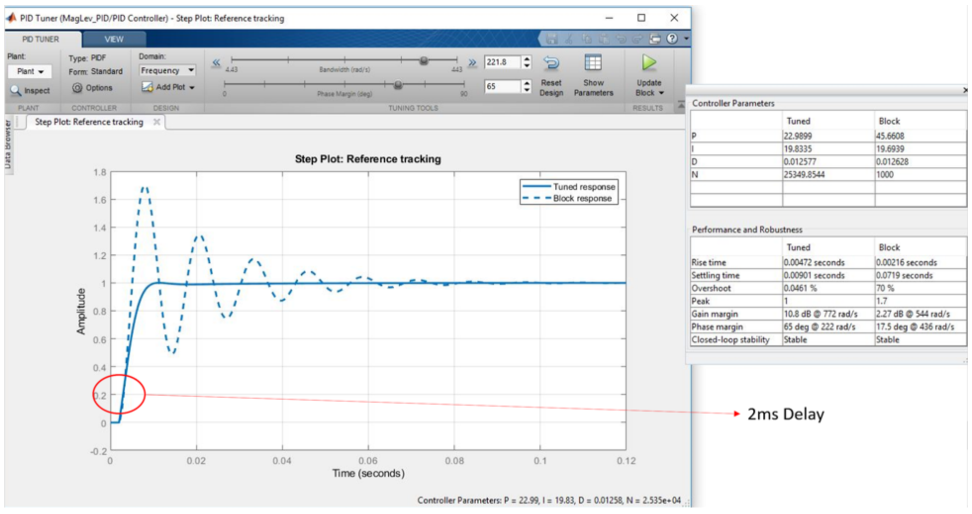


Figure 16: The tuned system's Nyquist plot is shown here with no encirclements about $(-1 + 0j)$ which indicates stability. The encirclements penetrate the negative real axis at 31 rad/s, corresponding to the system's resonance frequency, although stability is still maintained here.

To further replicate a realistic system, 2ms input signal and controller delays were added to the model as shown in Figure 10, and new correspondent PID values were tuned as shown in Figure 17; this delay value was chosen after examining the testing rig. The tuning that ensued was crucial to the procurement of appropriate PID values, as the inclusion of delays can greatly impact system performance (notice the 70% overshoot and 70ms settling time in the figure below). The re-tuned system was shown to be stable for delays of 2ms and a controller sampling rate of 1kHz.



Controller Parameters

	Tuned	Block
P	22.9899	45.6608
I	19.8335	19.6939
D	0.012577	0.012628
N	25349.8544	1000

Performance and Robustness

	Tuned	Block
Rise time	0.00472 seconds	0.00216 seconds
Settling time	0.00901 seconds	0.0719 seconds
Overshoot	0.0461 %	70 %
Peak	1	1.7
Gain margin	10.8 dB @ 772 rad/s	2.27 dB @ 544 rad/s
Phase margin	65 deg @ 222 rad/s	17.5 deg @ 436 rad/s
Closed-loop stability	Stable	Stable

Figure 17: Re-tuning the system with a 2ms controller delay, which impacted the performance and demanded new PID values. The final system response is characterized by a 0.0461% overshoot and a 9ms settling time, although the system is within sub-5% discrepancy as early as 2ms from the actuation time.

7 Physical System Design

7.1 Electromagnets

Various electromagnets were wound and tested during the research stage while specifications of the system were being explored through simulations. Table 2 below shows the properties of each electromagnet that was made.

Table 2: Electromagnet Revisions

Magnet	**v1**	**v2**	**v3**	**v4**	**v5**
Number of turns	110	210	210	210	410
Core Diameter	10mm	12mm	12mm	16mm	15mm
Length	70mm	35mm	50mm	100mm	100mm
Core Material	Steel Bolt	Stainless Steel	Steel Bolt	Steel Bolt	Cast Iron
Wire Thickness	1.5mm	1.5mm	0.5mm	1.5mm	0.75mm
Suitable	No	No	No	Yes	Yes

Due to the limited budget of the project they were made using scrap materials and spare insulated copper wire. These were all wound by hand which added an extra level of inaccuracy. Although each property had an effect on the system, the suitability was ultimately determined by three key factors: number of turns, core diameter and wire thickness.

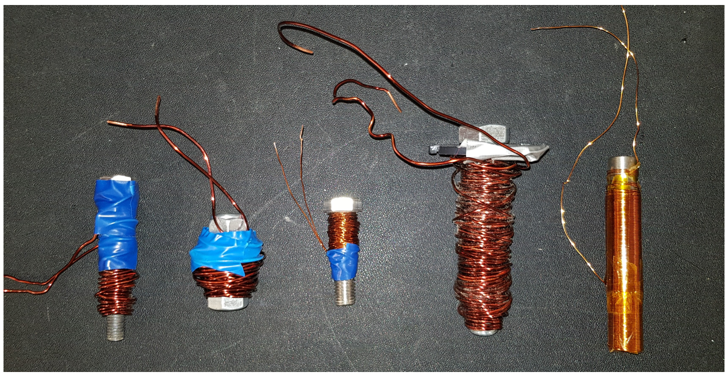


Figure 18: Electromagnets that were wound for testing

The first three versions were not suitable for three different reasons. V1 did not have enough turns to attract the ball, V2 did not have a ferromagnetic core and so did not magnetise, and V3 used wire that was too thin and had a very high internal resistance.

V4 and V5 performed well but ultimately V5 was used as it had the most accurately wound coils, used an iron core and showed the best results during testing.

7.2 Test Rig

The test rig serves three purposes to the overall system. It holds the magnet in place, allows for different sensors to be used, and provides easy adjustment of the levitation height. The rig was made from plastics to minimise its interaction with the magnetic circuit as that would interfere with the predicted system behaviour. Fortunately, 3D printers are abundant at the university, making construction from plastics easy and convenient.

Second to the functional aspects of the system, it is important that there is good visibility of the electromagnet and ball so that levitation can be monitored. Therefore a design featuring a top and bottom plate with four pillars connecting them together was used. The bottom was secured with M5 bolts while the top section used extruded plastic sections to connect to the base. This difference was to remove any pieces of conductive material from the magnetic circuit. Figure 19 shows the design of the final test rig that was used.

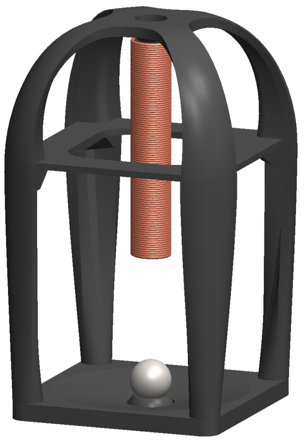


Figure 19: Test rig

Furthermore this test rig featured an adjustable mount so that the ball's distance could be easily adjusted which proved useful during the experimental phase.

8 Sensors

8.1 Photovoltaic Cell

The sensor that we elected to use for locating the ball was a photovoltaic (PV) cell. The PV cell was placed on one side of the rig and a light source on the other such that the ball being levitated was in between the two. As the ball started to levitate, it cast a larger shadow on the PV cell and hence decreased its output voltage.

8.1.1 Procurement

Due to the limited budget, purchasing either a PV cell or light source was outside the scope of the project. The PV cell was taken from a solar cell phone charger and the light source came from a small flashlight where the control circuitry was removed as it had a built in buck boost converter which meant that the output light was discontinous. The discontinuity would not have been a problem during intended use of the light as the frequency was too high to be observed, it caused significant problems for the sensor.

8.1.2 Sensor Readings

To get readings with as low a latency as possible, (given that the Raspberry Pi does not have the ability to read analogue signals) a protocol was developed for an Arduino where it would wait for an interrupt signal from the Raspberry Pi, when it received this signal, it would take a sensor reading and split the resultant voltage into 4 bytes for higher transfer speeds and then send it back to the Raspberry Pi. This allowed for the frequency of the controller to operate at around 2kHz and the latency be less about 1.5ms. With more work, this could have been further reduced but based on our theoretical models, this was deemed sufficient.

8.1.3 Challenges

There were two persistent challenges facing the use of the combination of PV cell and light source as a position sensor for the system. The first one was the issue of non-linearity in the sensor and the other was the noise in the sensor. These problems partially arose from the nature of analogue sensors but primarily from the fact that the equipment came from taking cheap electronics apart and was never intended to be used as a sensor.

Linearisation

Linearisation of the ball caused two main challenges, the first one was rather simple and was caused by the spherical shape of the ball. This meant that the rate at which the shadow grew as the ball moved was a function of its position. This could have been easily addressed by solving a differential equation of the form:

$$V \propto \frac{dV}{ds}$$

where V itself is a function of the light intensity, the geometric properties of the ball and its position. This would have been easily solvable if it were not for the second factor affecting the linearisation of the sensor.

This second factor is that the ball was not just moving in the y dimension but was also moving in the x and z directions as a result of bouncing from overshoots occurring during the tuning process. The fact that the ball was moving in the x direction (the axis on which both the light and PV cell were on) meant that the shadow would grow and shrink as the ball moved along that axis.

One possible solution to the linearsation problem would have been to make a small slit in the PV cell and cover the rest in an opaque material, this would have meant that the sensor could have, in effect, been linearised. However, as the ball was moving in the z direction, this was not an option for us.

In the end, the attempts at linearising the sensor all added an unacceptable sacrifice in reliability and robustness of the reading. As a result the non-linear sensor was used but this does present a scope for further research.

Noise

There were two primary sources of noise on the system, the first from the 100Hz frequency emitted by the ceiling lights. There was an attempt at electronic filtering of this using a simple low pass filter but the 100Hz was close enough to the operating frequency of the system to cause problems with the sensor signal. Therefore, a mechanical shielding system was devised instead for the purposes of prototyping (fig 20). This proved effective for blocking the signal from the ceiling lights but did obstruct the view of the ball thus making the setup impractical for demonstration purposes. This could have been overcome by a stronger light source or by adding a colour filter on the PV cell and using a coloured LED for the light source but these options were outside the scope of this project.

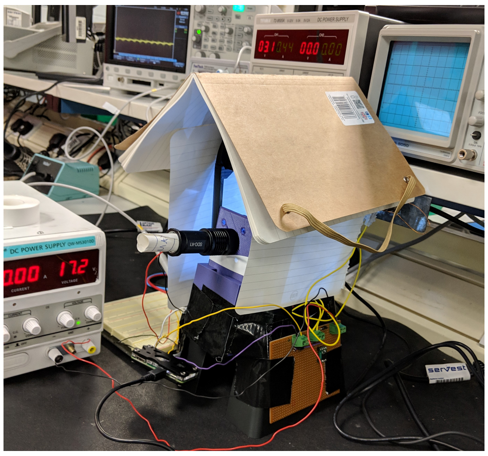


Figure 20: Shielding from lightbulbs 100Hz interference

The other significant problem that persisted throughout the project was the noise on the signal. Both the power supply and other electrical equipment in the lab supplied high frequency noise to the sensor, which, due to its low power output, was especially susceptible to this kind of noise, pushing the signal to noise ratio up to 10%. The problem was partially solved by wrapping the wires for the sensor in tinfoil, forming a makeshift Faraday cage, which blocked most of the electromagnetic interference. Additionally, the strength of the signal was boosted using a unity gain amplifier which further decreased the effect of the noise, down to less than 2%. However, this initially caused problems with the magnetic driver circuit overheating (see section 9).

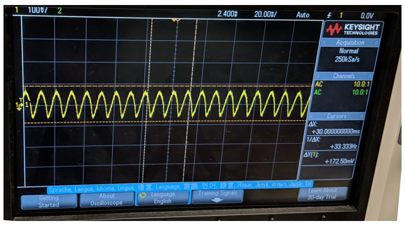


Figure 21: Unshielded signal without Amplifier

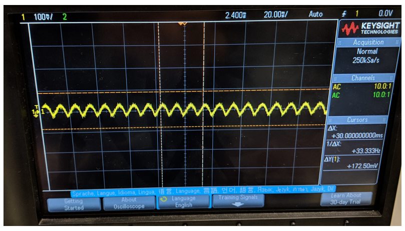


Figure 22: Shielded signal without Amplifier

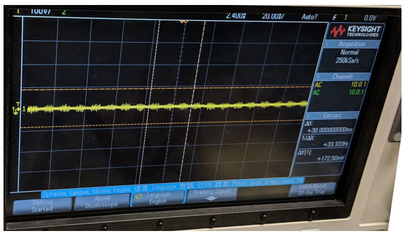
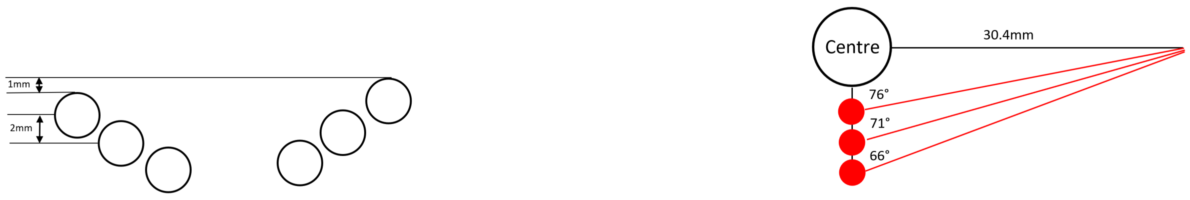


Figure 23: Shielded signal with Amplifier and filter

8.2 Lasers and Photodiodes

Pairs of lasers and photodiodes can be used to measure the height of the ball over a set range by focusing the beams of light at the centre of the test rig. In this implementation six lasers and six photodiodes were arranged into an offset pattern as shown in 24 (a) to achieve a range of 6mm and resolution of 1mm.



(a) Spacing (b) Angles

Figure 24: Geometry of laser system

The lasers were mounted inside angled slots with the geometry shown in 24 (b) so that they converged at the point where the ball would be moving. The photodiodes had a response time of 9ns but the readings had to go to an ADC so there was an overall 1ms response time. This could have been reduced by using a faster microcontroller or a lower level programming language such as C++.

8.3 Other Considered Sensor Options

8.3.1 Computer Vision

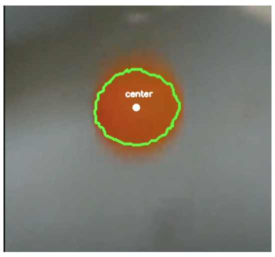


Figure 25: Still of computer vision tracking ball bearing

Computer vision had the potential to be an ideal option for this project; it offers accurate tracking of the ball and with implementation of stereo cameras has the potential to scale into both two and three dimensions of control. However, after some testing it was discarded for not having a quick enough read speed for our purposes.

There was an initial hope that the read rate could have been made high enough for computer vision to be usable for the project, and as a result, a lot of effort was put into making computer vision a viable option for the project. Creating an algorithm capable of identifying and tracking the ball was relatively easy to develop but the challenge came in being able to do this quickly enough to make computer vision a useful sensor. The initial approach for tracking the ball was to use a combination of a colour filter and a gradient based approach where the colour filter would initially identify the ball and the gradient method was used to follow it as it moved. This however turned out to take too long and would have problems with smudging of the picture when the ball was moving fast. Instead, the gradient filter was abandoned and replaced by a 'centre of mass' calculation algorithm of the ball based soley on the colour filter. (The colour filter calculated the proportion of orange within each pixel). The centre of mass calculations proved much more accuracy, as the smudging was roughly equal on top and bottom parts of the ball and, as can be seen in Figure 25, although the algorithm cannot exactly estimate the shape of the ball, it still finds the centre of it accurately and this is what was important to us.

Although this approach seemed promising, we soon learned that we needed a sample rate of at least 200Hz and a delay of less than 5ms, and although 200 fps are within reach of high end cameras and a 5ms delay is within the reach of most computers given a sufficiently optimised algorithm, we did not have access to such a camera and as this project is focused on engineering rather than computer science, we did not think it worth investing time in optimising an algorithm to that point.

8.3.2 Photoresistor

Photoresistors are, as the name implies, resistors that are sensitive to light. The resistance of the sensor changes as the light incident on the sensor changes. This means that we could have placed a number of them in an array and positioned them vertically on one side of the ball bearing with a light source on the other side. Then we would be able to tell whether the ball was in front of a particular sensor as it would cast a shadow on the sensor, thereby letting us track its location. Additionally, they are very cheap and simple to use. Furthermore, for the operational read speeds that we were intending, their innate capacitance is insignificant which could have made them ideally suited for this operation but unfortunately, the ones available to us were far too large meaning that they would not be able to accurately enough track the position of the ball bearing.

8.3.3 Ultrasonic Sensor

Another sensor that was considered was an ultrasonic distance sensor. It would have been cheap and easy to work with but it was discarded before we even started to figure out how to place it, as the datasheet verified that it was not accurate enough nor did it have an acceptable sample rate. It was therefore discarded at the stage of reading datasheets

8.3.4 Charged Coupled Device

Similar to the sensor of a camera, a charged couple device (CCD) relies on the photoelectric effect to build up an electric field in a capacitor or series of capacitors to measure the light incident on it over a period of time (until the capacitor is discharged). This type of sensor would have been ideal for us as it had given us very high accuracy at a sample rate that we ourselves could have specified. Unfortunately, a sensor of the size that we would have needed proved too costly for the purpose of the project. However, in any future industrial application, this sensor is the most appropriate among those that we have tested so far.

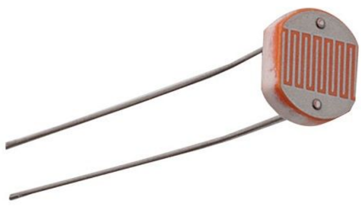


Figure 26: Photoresistor

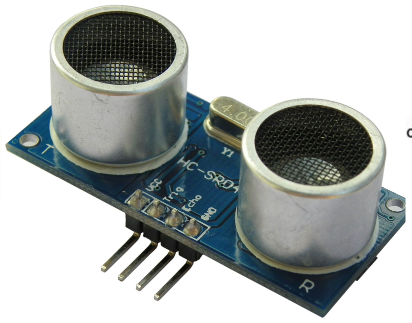


Figure 27: Ultrasonic transducer

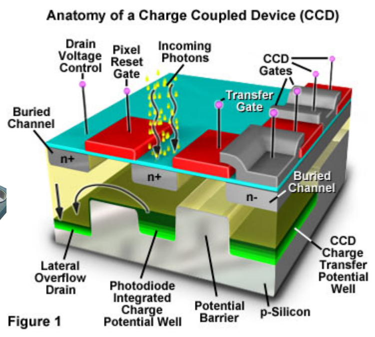


Figure 28: CCD

9 Control Circuit

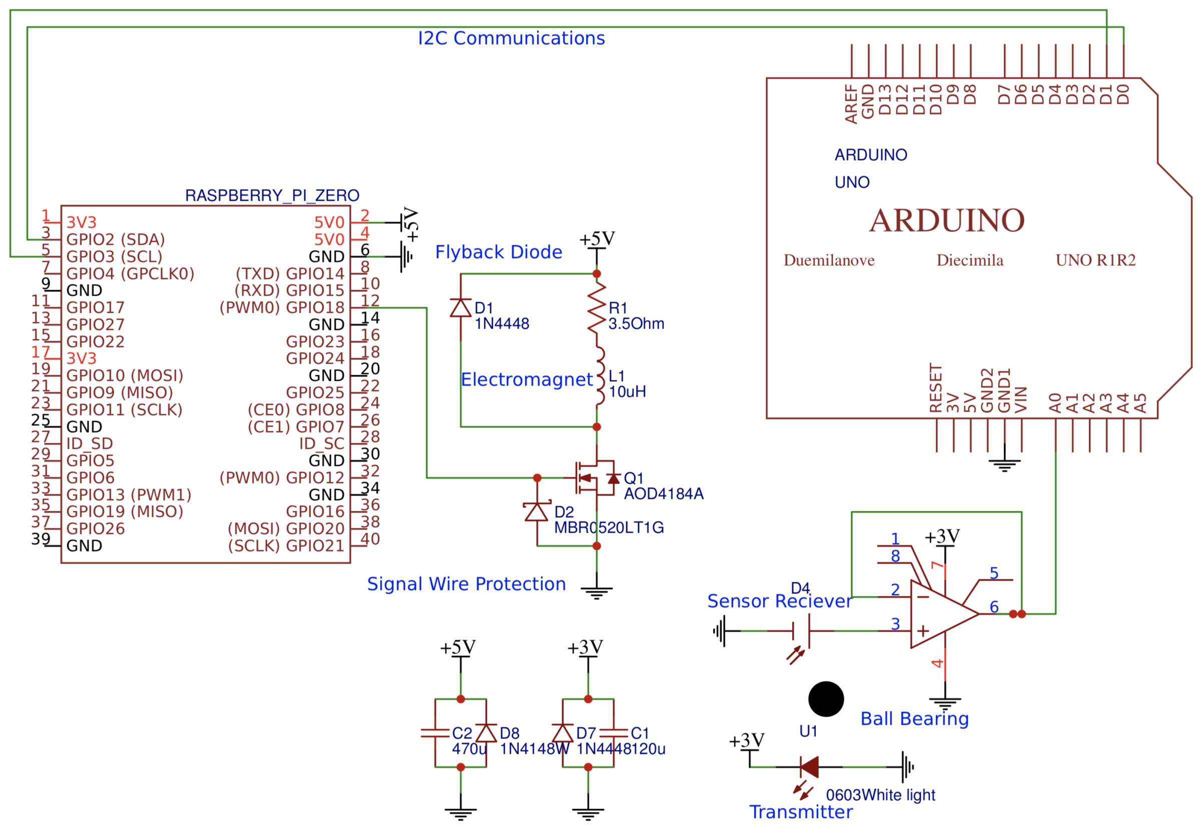


Figure 29: Magnetic control circuit

The control of the electromagnet was done with a single N-Channel MOSFET in a setup very similar to a step down converter where the N-channel was chosen on the basis of availability in the lab. Worthy of note is that reverse polarity protection diode were added to both the signal input as well as the power input, neither of which turned out to be unnecessary precautions.

There was a general concern that the electromagnet would magnetise it's own core which could have had significant impacts on system performance which would have necessitated the supply of AC current to the magnet instead of DC. This would have significantly complicated the circuitry as an inverter would have had to be made. Additionally, this would have made operations of the magnet less smooth as the AC current would oscillate the magnetic each cycle. These concerns proved to be unwarranted, however, as the magnetisation of the core proved to be significantly less that feared and seemed to not significantly affect the ball at all once the current was turned off.

A switching frequency of 10kHz was used to ensure a continuous current in the electromagnet, smoothing the operation and preventing voltage spikes.

One issue that appeared with the the MOSFET, when sealed inside the metal box to block out noise, was the fact that the MOSFET overheated. This was not unexpected as a lot of current was moving through it and was solved by adding a heatsink.

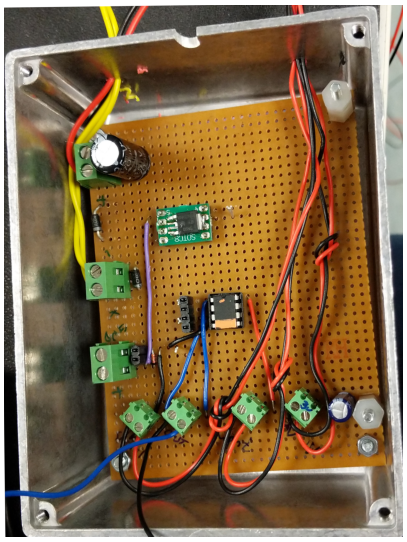


Figure 30: Case without Heatsink

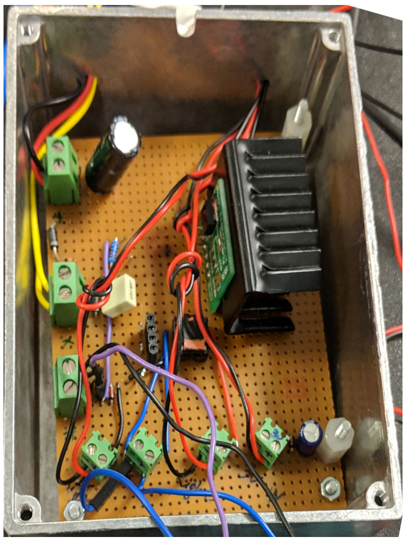


Figure 31: Case with Heatsink

10 Software Control and Processing

Two factors were the main drivers behind the development of the software, the first one was runtime; anything added to the code had to be carefully considered as this could increase the processing time of each measurement and therefore decrease the read frequency and the latency. This became especially evident when implementing logging features to save the performance of the system as they initially substantially increased the processing time. This, however, was fixed with some optimisation and proved to not be an issue. The other main driving factor was the manipulation of the input signals; PID controllers work best when working on linear systems with linear sensors. Two challenges arose from this. There was an initial attempt to process the input data. Since the ball is spherical, it did not cause an even shadow on the PV cell as it rose up, creating a non-linear sensor reading (as mentioned in section 8.1.3. There was an attempt at modelling the shape function of the shadow cast on the PV cell but there were too many unknowns for the attempted functions to work properly. In the end, the sensor was modelled as a simple parabola ($V \propto s^2$). Although not very accurate, it was more robust to 3 dimensional movement compared to the differential equation approached initially tried.

The other piece of adjustment that needed to be done was to the actuator controller. As shown in Figure: 3 the actuation force is proportional to the square of the distance from the electromagnet. As a PID controller is meant to control linear systems where a constant input to the actuator yields a constant force. In order to compensate for this, we added a function in the software after the PID with the following functionality:

$$G = G_{PID} \times d^2_{normalised}$$

The exact effects of this were difficult to measure due to a lack of equipment but it significantly improved system performance.

One more piece of software filtering that was added to the system was a gain limiter preventing the gain from exceeding 1; as the PID drives a PWM controller, it cannot be set to a larger value than 100%. This does diverge from an ideal PID controller but cannot be avoided with our setup.

11 Derivation of the 2D Transfer Function

After testing the 1D magnetic levitation system, the theoretical model was extended from 1D to 2D. Based on our model and assumptions, the ball could move from the target point to any near position. In addition, this method could also be implemented in 3D movement without much modification.

11.1 Physical Model and Assumptions

The 2D system can be modelled as shown in Figure 32. It consists of two identical electromagnets and a steel ball. The positive directions for x and y are also identified.

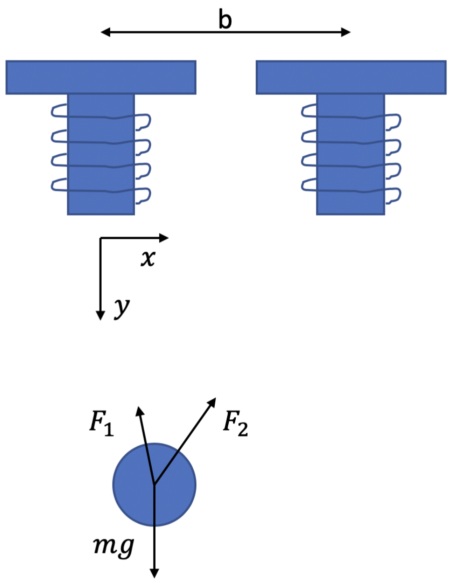


Figure 32: 2D magnetic levitation ball model.

Several assumptions were made in finding the 2D transfer functions. Firstly, all the assumptions made in 1D model were still valid in the 2D model. Secondly, it was assumed that the y distance between the ball and the solenoids was much greater than the x distances between the ball and two solenoids. Hence the distance between the ball and the solenoids could be simplified as the value of y. Thirdly, the direction of the magnetic force was assumed to be along the line between the centre of the ball and the centre of the lower end of the electromagnetic. In addition, a small displacement assumption was made so that the directions of F_1 and F_2 would not change during the movement. This was important in deriving the transfer function, because it was hard to deal with a term inside the root at the denominator.

The inductance was approximated as first order accurate (Hlebowitsh, 2012) as

$$L(y) \approx L_0 + \frac{L_1}{1 + ay} \tag{11.1}$$

where $1/a$ is the characteristic length.

Then, the coenergy of the system (Roberge, 1975) could be expressed as

$$E = \frac{1}{2} L(y) I^2 \tag{11.2}$$

Thus the force was

$$F = \frac{\partial E}{\partial y} = \frac{-aL_1 I^2}{2(1 + ay)^2} \tag{11.3}$$

If the operating point was chosen at $y = y_0$, the current at that point was I_0 and the magnetic force generated at that point was F_0, we could linearise the force formula at this point (Hlebowitsh, 2012) as

$$F = F_0(I_0, y_0) + \frac{\partial F}{\partial y} dy + \frac{\partial F}{\partial I} dI = \frac{-aL_1 I_0^2}{2(1 + ay_0)^2} + \frac{a^2 L_1 I_0^2}{(1 + ay_0)^3} dy - \frac{aL_1 I_0}{(1 + ay_0)^2} dI \tag{11.4}$$

where $I = I_0 + dI$ and $y = y_0 + dy$.

Due to the assumption of small displacement, the vertical balance function at the operating point could be written as

$$0 = mg + F_1(I_0, y_0) \cdot \frac{y_0}{\sqrt{x_0^2 + y_0^2}} + F_2(I_0, y_0) \cdot \frac{y_0}{\sqrt{(b - x_0)^2 + y_0^2}}$$

$$\Rightarrow 0 = mg + \frac{-aL_1 I_1^2}{2(1 + ay_0)^2} \cdot \frac{y_0}{\sqrt{x_0^2 + y_0^2}} + \frac{-aL_1 I_2^2}{2(1 + ay_0)^2} \cdot \frac{y_0}{\sqrt{(b - x_0)^2 + y_0^2}} \tag{11.5}$$

At the same time, the horizontal portion of the magnetic force from two solenoids must balance each other. This lead to

$$F_1(I_0, y_0) \cdot \frac{x_0}{\sqrt{x_0^2 + y_0^2}} = F_2(I_0, y_0) \cdot \frac{b - x_0}{\sqrt{(b - x_0)^2 + y_0^2}}$$

$$\Rightarrow \frac{-aL_1 I_1^2}{2(1 + ay_0)^2} \cdot \frac{x_0}{\sqrt{x_0^2 + y_0^2}} = \frac{-aL_1 I_2^2}{2(1 + ay_0)^2} \cdot \frac{b - x_0}{\sqrt{(b - x_0)^2 + y_0^2}} \tag{11.6}$$

If we assumed $\sqrt{x^2 + y^2} \approx \sqrt{(b - x)^2 + y^2}$ ($y \gg x$ and $y \gg (b - x)$), then we could obtain

$$I_1^2 \cdot x_0 = I_2^2 \cdot (b - x_0) \tag{11.7}$$

In addition, we also assumed all the coefficients so that we could calculate the example transfer functions.

$$a = 1\,m^{-1} \quad m = 1\,kg \quad L_0 = 1\,H \quad L_1 = 0.1\,H$$

$$y_0 = 0.1\,m \quad x_0 = 0.005\,m \quad b = 0.02\,m$$

To maintain the ball at this position, Eq.11.5 and Eq.11.6 could be solved to find

$$I_1 = 13.3A \quad I_2 = 7.7A$$

11.2 Vertical Movement

The 2D movement could be divided into two steps of vertical movement and horizontal movement. We could write the vertical movement equation for the steel ball with small displacement from the operating point as

$$\begin{aligned} m\frac{d^2y}{dt^2} &= [\frac{a^2L_1I_1^2}{(1+ay_0)^3}dy - \frac{aL_1I_1}{(1+ay_0)^2}dI_1] \cdot \frac{y_0}{\sqrt{x_0^2+y_0^2}} \\ &+ [\frac{a^2L_1I_2^2}{(1+ay_0)^3}dy - \frac{aL_1I_2}{(1+ay_0)^2}dI_2] \cdot \frac{y_0}{\sqrt{(b-x_0)^2+y_0^2}} \end{aligned}$$

$$\begin{aligned} \Rightarrow m\frac{d^2y}{dt^2} &= [\frac{a^2L_1I_1^2}{(1+ay_0)^3} \cdot \frac{y_0}{\sqrt{x_0^2+y_0^2}} + \frac{a^2L_1I_2^2}{(1+ay_0)^3} \cdot \frac{y_0}{\sqrt{(b-x_0)^2+y_0^2}}]dy \\ &+ [-\frac{aL_1I_1}{(1+ay_0)^2} \cdot \frac{y_0}{\sqrt{x_0^2+y_0^2}}]dI_1 + [-\frac{aL_1I_2}{(1+ay_0)^2} \cdot \frac{y_0}{\sqrt{(b-x_0)^2+y_0^2}}]dI_2 \end{aligned} \tag{11.8}$$

If we had constants of

$$k_y = \frac{a^2L_1I_1^2}{(1+ay_0)^3} \cdot \frac{y_0}{\sqrt{x_0^2+y_0^2}} + \frac{a^2L_1I_2^2}{(1+ay_0)^3} \cdot \frac{y_0}{\sqrt{(b-x_0)^2+y_0^2}} \tag{11.9}$$

$$k_{i1} = -\frac{aL_1I_1}{(1+ay_0)^2} \cdot \frac{y_0}{\sqrt{x_0^2+y_0^2}} \tag{11.10}$$

$$k_{i2} = -\frac{aL_1I_2}{(1+ay_0)^2} \cdot \frac{y_0}{\sqrt{(b-x_0)^2+y_0^2}} \tag{11.11}$$

the equation became

$$m\frac{d^2y}{dt^2} = k_ydy + k_{i1}dI_1 + k_{i2}dI_2 \tag{11.12}$$

It is noted that these terms are constants exclusively in the case of small displacement. However, for a larger displacement, an accurate sensor could be used to measure the position of the ball in a high frequency and recalculate these constants so that the small displacement assuption could be always valid.

After Laplace Transform, this equation became

$$ms^2dy = k_y dy + k_{i1}dI_1 + k_{i2}dI_2 \tag{11.13}$$

Hence the transfer functions could be found to be

$$\frac{\Delta Y(s)}{\Delta I_1(s)} = \frac{k_{i1}}{ms^2 - k_y} \tag{11.14}$$

$$\frac{\Delta Y(s)}{\Delta I_2(s)} = \frac{k_{i2}}{ms^2 - k_y} \tag{11.15}$$

If we input all the coefficients assumed in the previous section, the transfer function would become

$$G_{PYi1} = \frac{\Delta Y(s)}{\Delta I_1(s)} = \frac{-1.10}{s^2 - 17.68} \tag{11.16}$$

$$G_{PYi2} = \frac{\Delta Y(s)}{\Delta I_2(s)} = \frac{-0.64}{s^2 - 17.68} \tag{11.17}$$

Based on the balance of horizontal forces, a relationship between the change of I_1 and I_2 could also be built as

$$\frac{\Delta I_1(s)}{\Delta I_2(s)} = \sqrt{3} \tag{11.18}$$

at the operating point mentioned before. Thus two transfer functions could be combined as one if that relationship was always kept. For example, the vertical movement controlled by single current I_1 would be

$$\frac{\Delta Y(s)}{\Delta I_1(s)} = \frac{-1.47}{s^2 - 17.68} \tag{11.19}$$

11.3 Horizontal Movement

We could also write the horizontal movement equation for the steel ball with small displacement from the operating point as

$$\begin{aligned} m\frac{d^2x}{dt^2} &= [-\frac{aL_1I_1dI_1}{(1+ay)^2}] \cdot \frac{x_0}{\sqrt{x_0^2+y_0^2}} + \frac{aL_1I_2dI_2}{(1+ay)^2} \cdot \frac{b-x_0}{\sqrt{(b-x_0)^2+y_0^2}} \\ \Rightarrow m\frac{d^2x}{dt^2} &= [-\frac{aL_1I_1}{(1+ay)^2}] \cdot \frac{x_0}{\sqrt{x_0^2+y_0^2}}dI_1 + \frac{aL_1I_2}{(1+ay)^2} \cdot \frac{b-x_0}{\sqrt{(b-x_0)^2+y_0^2}}dI_2 \end{aligned} \tag{11.20}$$

Then, we could introduce two constants of

$$k_{i1} = [-\frac{aL_1I_1}{(1+ay)^2}] \cdot \frac{x_0}{\sqrt{x_0^2+y_0^2}} \tag{11.21}$$

$$k_{i2} = \frac{aL_1I_2}{(1+ay)^2} \cdot \frac{b-x_0}{\sqrt{(b-x_0)^2+y_0^2}} \tag{11.22}$$

Then, the equation became

$$m\frac{d^2x}{dt^2} = k_{i1}dI_1 + k_{i2}dI_2 \tag{11.23}$$

After Laplace Transform,

$$ms^2x = k_{i1}dI_1 + k_{i2}dI_2 \tag{11.24}$$

Thus the transfer functions were

$$\frac{\Delta X(s)}{\Delta I_1(s)} = \frac{k_{i1}}{ms^2} \tag{11.25}$$

$$\frac{\Delta X(s)}{\Delta I_2(s)} = \frac{k_{i2}}{ms^2} \tag{11.26}$$

If the assumed coefficients were input into the equation, the transfer functions would be

$$G_{PXi1} = \frac{\Delta X(s)}{\Delta I_1(s)} = \frac{-0.055}{s^2} \tag{11.27}$$

$$G_{PXi2} = \frac{\Delta X(s)}{\Delta I_2(s)} = \frac{0.094}{s^2} \tag{11.28}$$

In horizontal movement, the vertical force balance should always be kept, which lead to

$$\frac{aL_1I_1}{(1+ay_0)^2}dI_1 + \frac{aL_1I_2}{(1+ay_0)^2}dI_2 = 0$$

$$\Rightarrow I_1dI_1 + I_2dI_2 = 0$$

$$\Rightarrow \frac{\Delta I_1(s)}{\Delta I_2(s)} = -\frac{\sqrt{3}}{3} \tag{11.29}$$

Hence, the horizontal movement could be controlled by single current I_1 as

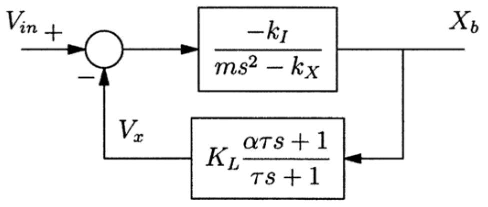
$$\frac{\Delta X(s)}{\Delta I_1(s)} = \frac{-0.22}{s^2} \tag{11.30}$$

12 2D Control System Design

The most significant elements of this projects expansion into 2D control are the emergence of coupled transfer functions and further emphasis on nonlinearity for the multiple-input/multiple-output (MIMO) system. This exercise of strategizing a new control method is a theoretical one, as budget constraints for this project do not permit the full exploration and validation of the assumptions and simplifications made in the transfer function derivation done above. Hence, the proposed control strategy will attempt to account for these assumptions to emulate a realistic system.

Given the advanced scope of this system, the first step was to explore control methods attempted by reputable sources. A satisfactory implementation was written by Hlebowitsh (2012); the emphasis of nonlinearity in the system demands a controller that will employ fast and subtle changes as opposed to slow changes with larger effects. The Lead-Lag Compensator satisfies this criterion, and is employed using variables within the transfer function, as no reasonable assumptions can be made regarding any individual component for our theoretical system.

In the application proposed, the writers suggest using lead-lag compensation in the feedback path to tackle the nonlinearity of the system, which is done as follows by targeting the first axis of control:



V_{in}
+
−
$\frac{-k_I}{ms^2 - k_X}$
X_b
V_x
$K_L \frac{\alpha\tau s + 1}{\tau s + 1}$

Figure 33: Lead compensator in the feedback path for control of the horizontal dimension as proposed by Hlebowitsh (2012).

The resultant system is characterised by:

$$\begin{aligned}
\frac{X_b(s)}{I_m(s)} &= \frac{\frac{-k_1}{ms^2-k_X}}{\frac{-k_I}{ms^2-k_X} K_L \frac{\alpha\tau S+1}{\tau s+1} + 1} \\
&= \frac{-k_I}{-k_I K_L \frac{\alpha\tau+1}{\tau s+1} + ms^2 - k_X} \\
&= \frac{-k_I(\tau s + 1)}{-k_I K_L(\alpha\tau s + 1) + (ms^2 - k_X)(\tau s + 1)}
\end{aligned} \tag{12.1}$$

As the determination of the stability of this equation is laborious, the root locus method is used instead, where the loop gain can be expressed as:

$$\frac{-k_I}{ms^2 - k_X} K_L \frac{\alpha\tau s + 1}{\tau s + 1} \tag{12.2}$$

This identifies system poles at $\pm\sqrt{\frac{k_x}{m}}$ and $\frac{-1}{\tau}$, with a zero at $\frac{-1}{\alpha\tau}$.

The resultant root locus plot is shown here

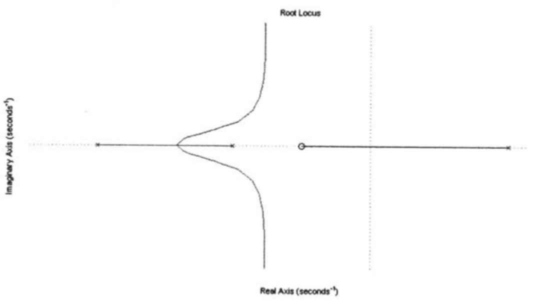
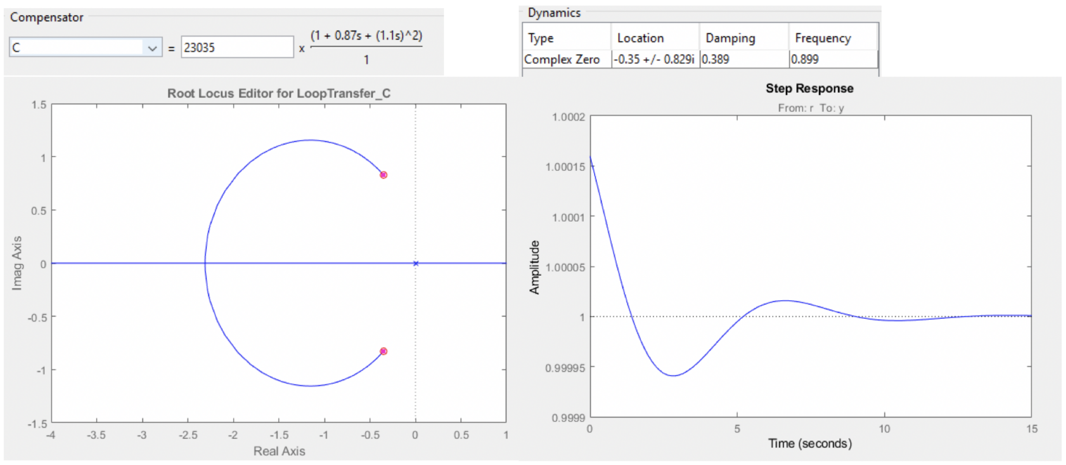


Figure 34: Root locus plot by Hlebowitsh (2012)

At $K_L = 0$, the poles of the open-loop poles of the transfer function, G(s)H(s) are the starting point of the root locus plot. These are denoted by an "x" in the figures 34 and 35. As K_L increases, the poles begin moving towards the zeros of the open-loop transfer function, $G(s)H(s)$. These are denoted by an 'o' in figures 34 and 35.

Hence, it can be seen from the plot that the lead compensator provides stability to the nonlinear system by moving the extraneous pole $\sqrt{\frac{k_x}{m}}$ into the negative real axis for high gains. Thus, it is a suitable controller. Crucially, as the transfer function for the 2D system can be described similarly, albeit with different constants, this control method is applicable there as well. Here is the tuning result using a complex pole for one our horizontal component controlled by solenoid 1 shown as an exercise. This uses arbitrary values for the transfer function variables:


Compensator

C = 23035 x (1 + 0.87s + (1.1s)^2) / 1

Dynamics

Type	Location	Damping	Frequency
Complex Zero	-0.35 +/- 0.829i	0.389	0.899

Figure 35: Root locus plot where a compensator has been implemented, and the gain has been increased to result in stable behaviour with a settling error of 0.015%. The poles have moved to the location of the zeros with an increase in Gain.

The control strategy proposed by this Hlebowitsh (2012), however, is at odds with this project's goal of 2D levitation. Their proposed system is not described in detail, so the descriptive control method must be inferred from their diagrams; here is their control model:

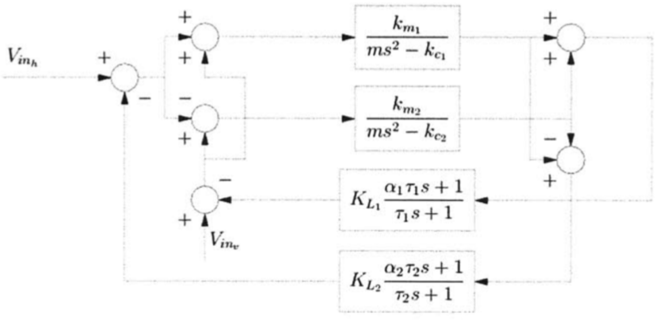

V_{in_h}
V_{in_v}
$\frac{k_{m_1}}{ms^2 - k_{e_1}}$
$\frac{k_{m_2}}{ms^2 - k_{e_2}}$
$K_{L_1}\frac{\alpha_1\tau_1 s+1}{\tau_1 s+1}$
$K_{L_2}\frac{\alpha_2\tau_2 s+1}{\tau_2 s+1}$

Figure 36: Proposed control model by (Hlebowitsh, 2012)

They argue that by limiting the vertical displacement to near-zero, they are able to provide stable control of the system in the horizontal direction. Hence, although their control strategy is one of two axes, as there is control over the vertical displacement, the effective result is single-axis motion in the horizontal direction. This means that their setup must be calibrated for a static vertical component for each demonstration.

Each of their transfer functions is simplified to discard its respective coupled component so that it can instead be implemented via superposition within the control model itself; that is to say, the horizontal transfer function discards the influence of the balls vertical position, and vice versa. Furthermore, the same current is used to control both solenoids. This is a good method of linearizing the model, but the implementation of feedback is limiting.

It can be seen from their control diagram that the model sums the effect of each solenoid in the 2-solenoid setup to account for the vertical force on the ball and subtracts the effect of the second solenoid from the first to account for the horizontal force on the ball. This is accurate in relation to the free-body-diagram. However, in the feedback path, their reliance on a single current to control both solenoids causes complications.

This approach only works with the assumption that either the horizontal or the vertical displacement can be set to zero and is hence negligible; they have chosen the latter to be set to zero in this case and have thereby linearized their model. Hence, they system implements '2D' control as it considers both components of motion, but the effective result of motion is still 1-dimensional, as the fundamental focus of the contorller is to limit vertical movement to allow for horizontal movement using a linearized model.

The following control strategy proposes to eliminate this limitation by controlling each solenoid with a separate current, as shown in the transfer function derivations in the previous section. The method employed is inspired by the strategy described above, with the difference being that it is improved to provide '2D' movement, not just '2D' control. This is done by implementing a switching control method, whereby to move the ball to the target point, the actuation is done in small linearized steps. As shown in Figure 37, each iterative step, if made sufficiently small, - as known to be possible using a lead-lag compensator - can disregard the effect of its coupling nonlinear terms. Hence, it is possible to validate the linearization by iteratively changing the system's equilibrium point and controlled variable throughout the course of the ball's motion. The fundamental forseeable limitation in this approach is the high computing power required to measure and actuate such small steps, as larger steps are more susceptible to the effects of nonlinearity. The high-level description of this strategy is shown below:

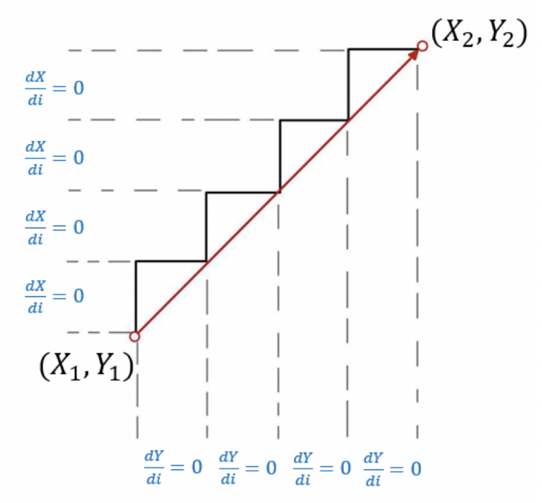


Figure 37: Iterative step motion strategy. During horizontal movement, the vertical component is unchanged. During vertical movement, the horizontal component is unchanged

This new control strategy can be described by the MIMO system shown below, where compensators for i1 and i2 are inferred to be similar to those of a 1D system, as was done by (Hlebowitsh, 2012).

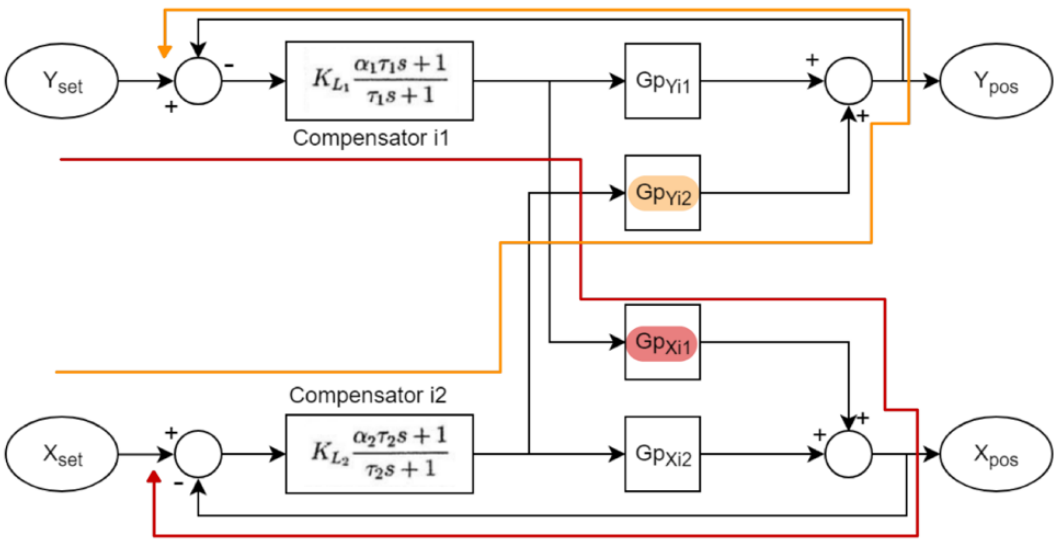

Yset
$K_{L_1}\frac{\alpha_1\tau_1 s+1}{\tau_1 s+1}$
Compensator i1
GpYi1
GpYi2
GpXi1
Ypos
Compensator i2
Xset
$K_{L_2}\frac{\alpha_2\tau_2 s+1}{\tau_2 s+1}$
GpXi2
Xpos

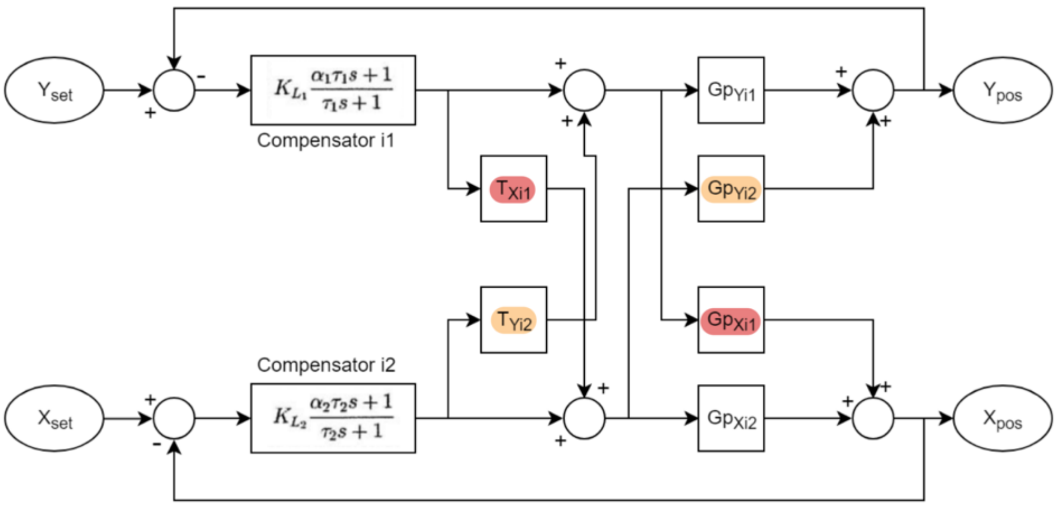
Figure 38: $Gp_Y i1$ refers to the plant model describing the relationship between the vertical displacement and the current from solenoid 1. Similarly, Gp_{Yi2}, Gp_{Xi1}, and Gp_{Xi2} refer to their respective plant models.

Here, the effects of each solenoid on each direction of displacement are described by these plant models, and they are summed to account for the combined effects of both solenoids. Gp_{Xi1} and Gp_{Xi2} can also be summed as the opposed effect they have on the horizontal displacement of the ball is already accounted for in their respective transfer functions. This is in the interest of simplifying the decoupling process described later in this section. To allow for the linearization and independence of each transfer function, the MIMO system description is employed to account for the coupling effect of the vertical position on the horizontal motion (and vice versa) via indirect feedback loops. Here, the red line shows the indirect feedback path caused by the vertical displacement error on the horizontal controller, and similarly, the orange line shows the indirect feedback path caused by the horizontal displacement error on the vertical controller; this is a coupled MIMO system.

Finally, to allow for the independent tuning of each compensator, it is necessary to decouple the effect of i1 on horizontal motion and i2 on vertical motion. Linearized decoupler functions are described using the following:

$$T_{Yi2} = -\frac{Gp_{Yi2}}{Gp_{Yi1}} \quad T_{Xi1} = -\frac{Gp_{Xi1}}{Gp_{Xi2}} \tag{12.3}$$

This method is illustrated in Figure 39. Notably, it should retain the effect of the vertical position on horizontal control, and vice versa.



Y_set
X_set
Compensator i1
Compensator i2
$K_{L_1}\frac{\alpha_1 \tau_1 s+1}{\tau_1 s+1}$
$K_{L_2}\frac{\alpha_2 \tau_2 s+1}{\tau_2 s+1}$
T_Xi1
T_Yi2
Gp_Yi1
Gp_Yi2
Gp_Xi1
Gp_Xi2
Y_pos
X_pos

Figure 39: MIMO system with decoupler functions $T_{Yi2} = -\frac{Gp_{Yi2}}{Gp_{Yi1}}$ $T_{Xi1} = -\frac{Gp_{Xi1}}{Gp_{Xi2}}$

Here is the mathematical proof for this claim, where U1 and U2 are the control outputs from the compensators for selenoids 1 and 2 respectively:

Eliminating G_{pXi1}:

At the first summation block:

$$(T_{Xi1} \cdot U_1 + U_2)G_{pXi2} = -G_{pXi1}U_1 + G_{pXi2}U_2 \tag{12.4}$$

At the second summation block:

$$G_{pXi1}U_1 + G_{pXi2}U_2 - G_{pXi1}U_1 = G_{pXi2}U_2 \tag{12.5}$$

Eliminating G_{pYi2}:

At the first summation block:

$$(T_{Yi2} \cdot U_2 + U_1)G_{pYi1} = -G_{pYi2}U_2 + G_{pYi1}U_1 \tag{12.6}$$

At the second summation block:

$$G_{pYi2}U_2 + G_{pYi1}U_1 - G_{pYi2}U_2 = G_{pYi1}U_1 \tag{12.7}$$

A linearized decoupler is generally a gain, and is an effective method of resolving a MIMO system; it theoretically allows for the resultant 2D control of the system, as a high gain will enable the nullification of small errors throughout the step movement of the ball from its original point to its target, provided the sampling rate of the controller is high enough. This was shown in the analysis of the lead-lag compensator. The method presented here may call for the implementation of an analogue controller as opposed to the digital one provided for the 1D rig.

One potential issue with this linearized method is in the decouplers themselves. By introducing linear decoupler functions, it is possible that the system description was oversimplified, and that infeasibly powerful measurement and actuation hardware is needed to meet the requirements of iterative stepwise motion. Hence, it is noted that future investigations must be centered on implementing 2-degree-of-freedom (DOF) plants, controllers, and decouplers with both x and y inputs for a more accurate system description. Although data is needed to assess the validity of this issue, it may be nevertheless be necessary to implement nonlinear decoupling for this application. This may prove true if one considers the effects of magnetic remanence and hysteresis, which have not been described by the transfer functions to begin with. However, if shown to work, this model can easily be developed to describe a '3D' system thanks to the expandable scope of MIMO system descriptions.

13 Conclusion

13.1 System Implementation

In the end, the system did not function reliably, it was on the verge of stable operation on several occasions but there were simply too many unknowns for such a complicated system so the results could not be repeated successfully. The system suffered from several problems from noise to accuracy and precision of sensors, all of which could have been solved but given the short timescales and lack of budget, there was not enough time to both solve these problems and afterwards tune the system. Despite this, the project was not a failure as the system was on the verge of working. During the literature review, no system was found that was levitating a steel ball (all the implementations featured lifting magnets which significantly ease the levitation and position measuring).

13.2 2D

Regarding the endeavour of implementing control in two dimensions, a wholly descriptive, albeit linearised model of the system was developed along with an accompanying decoupling control strategy. Although this is hypothesised to work, it relies on the assumption that our stepwise compensator control can be made subtle enough to permit the linearization of relevant transfer functions. Namely, the decouplers in their present state - being linear - are merely gains. To further build upon this paper, nonlinear decoupling should be investigated for more robust and realistic control. However, this strategy takes the endeavour further than those presented by all investigated sources. While the restrictive control of one of the axes places the system somewhere between a '1D' and '2D' application, the method proposed here presents a clear '2D' approach. The remaining and crucial variable to this assertion is whether it will work on a real system. Notably, if it does work, it will easily scalable into a 3D implementation.

13.3 Final Thoughts

Due to the larger than initially estimated challenge of making the 1D practical system work, there was less scope for comparison between theory and experiment that initially expected. Due to both the non-linearity of the system and its unstable equilibrium the equipment available proved insufficient to achieve stable levitation within the time constraints of the project. However, with the new theory around 2D magnetic levitation developed as a proof that it is feasible and the advances made by the experimental rig suggest that with proper equipment, implementation should be achievable. This provides an excellent scope for further study of magnetic levitation where permanent magnets are not an option.

References

Amaral, J.F.M., Tanscheit, R. and Pacheco, M.A.C. 2005. Tuning evolvable PID controllers through a clonal selection algorithm. *NASA/DoD Conference on Evolvable Hardware.* pp.30-33.

ANSYS. 2018a. *ANSYS Maxwell.* [Online]. [Accessed 18 November 2018]. Available from: https://www.ansys.com/en-gb/products/electronics/ansys-maxwell

ANSYS. 2018b. *ANSYS AIM.* [Online]. [Accessed 18 November 2018]. Available from: https://www.ansys.com/en-gb/products/3d-design/ansys-aim

Barie, W. and Chiasson, J. 1996. Linear and nonlinear state-space controllers for magnetic levitation. *International Journal of Systems Science.* **27**(11), pp.1153-1163.

Duriez, T., Brunton, S.L. and Noack, B.R. 2017. *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence.* Springer International Publishing.

ELEC FREAKS - "Ultrasonic Ranging Module HC - SR04" http://www.Elecfreaks.com - DATASHEET

Finite Element Method Magnetics. 2014. *Finite Element Method Magnetics.* [Online]. [Accessed 18 November 2018]. Available from: http://www.femm.info/wiki/HomePage

Gazdos, F., Dostal, P. and Marholt, J. 2011. Robust control of unstable systems: algebraic approach using sensitivity functions. *International Journal of Mathematical Models and methods in Applied Sciences.* **5**(7), pp.1189-1196.

General Resitors, "CDS Photo Resistors - PGM Series" - DATASHEET

Hajjaji, A.E. and Ouladsine, M. 2001. Modeling and Nonlinear Control of Magnetic Levitation Systems. *IEEE Transactions of Industrial Electronics.* **48**(4), pp.831-838.

Hlebowitsh, P.G. 2012. *The Road to Multi-Dimensional Magnetic Levitation: Realizing Two-Dimensional Control in Classical Feedback.* Master of Engineering Dissertation, Massachusetts Institute of Technology.

Kumar, P. and Minz, S. Control of Magnetic Levitation System Using PD and PID controller. *International Journal of Scientific Research and Education.* **4**(8), pp.5596-5602.

Kumar E, V. and Jerome, J. 2013. LQR based optimal tuning of PID controller for trajectory tracking of Magnetic Levitation System. *Procedia Engineering.* **64**, pp.254-264.

Overdijk, D.A., van de Wouw, N. and de Kraker, A. 2001. Alternative Methods in Spectral Factorization. A Modeling and Design Tool. *Journal of Applied Mathematics and Mechanics.* **81**(2), pp.140-144.

Pedersen, G.K.M. and Yang, Z. 2006. Multi-Objective PID-Controller Tuning for a Magnetic Levitation System using NSGA-II. *Proceedings of the 8th annual conference on Genetic and evolutionary computation.* pp.1737-1744.

Roberge, J.K. 1975. *Operational Amplifiers: Theory and Practice.* [Online]. Massachusetts

Institute of Technology: MIT OpenCourseWare. [Accessed 25 Nov 2018]. Available from: https://ocw.mit.edu/resources/res-6-010-electronic-feedback-systems-spring-2013/textbook/

Schweitzer, G. and Maslen, E.H. 2009. *Magnetic Bearings: Theory, Design, and Application to Rotating Machinery.*Springer International Publishing.

Sorour, M.I. 2015. *Designing Two-Dimensional Magnetic Levitation Control System.* Doctor of Philosophy Thesis, The Islamic University-Gaza.

Woodson, H.H. and Melcher, J.R. 1968. *Electromechanical Dynamics.* [Online]. Massachusetts Institute of Technology: MIT OpenCourseWare. [Accessed 20 Nov 2018]. Available from: https://ocw.mit.edu/resources/res-6-003-electromechanical-dynamics-spring-2009/front-end-matter/

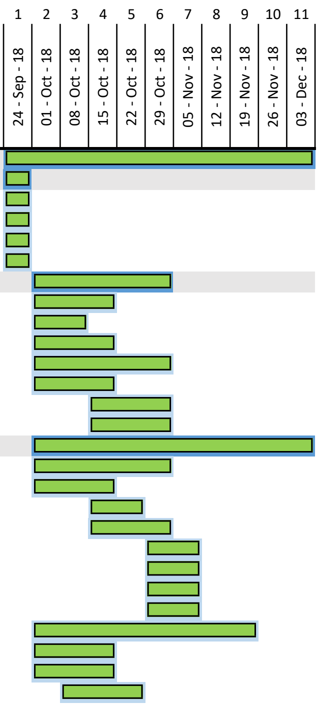
Appendix

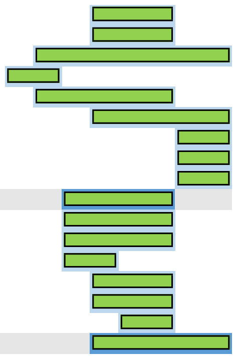
Gantt Chart

Project Plan

Project Manager:	Max Benson
Start Date:	24/09/2018
End Date:	29/11/2018

Section	Tasks	Task Leader	Start	End	Duration (Days)	% Complete	Days Remaining
0	**Overall Project**		24/09/18	29/11/18	66	100%	0.00
1	**Concept and Planning**		24/09/18	01/10/18	7	100%	0.00
1.1	Choose and define project		24/09/18	01/10/18	7	100%	0.00
1.2	Literature research		24/09/18	01/10/18	7	100%	0.00
1.3	High level design ideas		24/09/18	01/10/18	7	100%	0.00
1.4	Project plan	MB	24/09/18	01/10/18	7	100%	0.00
2	**1D Simulation**	MR / YX	01/10/18	31/10/18	30	100%	0.00
2.1	Development of Transfer Function	YX	01/10/18	15/10/18	14	100%	0.00
2.1.1	Magnetic simulation		01/10/18	08/10/18	7	100%	0.00
2.1.2	Derivation of stable transfer function		01/10/18	15/10/18	14	100%	0.00
2.2	Development of Control System	MR	01/10/18	31/10/18	30	100%	0.00
2.2.2	Control loop design (with feedforward)		01/10/18	15/10/18	14	100%	0.00
2.2.3	Graphical simulation of magnetic levitation		15/10/18	31/10/18	16	100%	0.00
2.2.4	Simulink model		15/10/18	31/10/18	16	100%	0.00
3	**1D Physical System**	MB / EH / DC	01/10/18	29/11/18	59	100%	0.00
3.1	Rig version 1	DC	01/10/18	31/10/18	30	100%	0.00
3.1.1	Procurement of parts		01/10/18	15/10/18	14	100%	0.00
3.1.2	Build electromagnet		15/10/18	22/10/18	7	100%	0.00
3.1.3	Assemble rig		15/10/18	31/10/18	16	100%	0.00
3.2	Rig version 2	DC	31/10/18	07/11/18	7	100%	0.00
3.2.1	CAD design and 3D printing of parts		31/10/18	07/11/18	7	100%	0.00
3.2.2	Build improved electromagnet		31/10/18	07/11/18	7	100%	0.00
3.2.3	Assemble rig		31/10/18	07/11/18	7	100%	0.00
3.3	Electrical design and implementation	EH	01/10/18	19/11/18	49	100%	0.00
3.3.1	Research and select sensor		01/10/18	15/10/18	14	100%	0.00
3.3.2	Specification for circuitry		01/10/18	15/10/18	14	100%	0.00
3.3.3	Design and build circuitry		08/10/18	22/10/18	14	100%	0.00
3.3.4	Reduce noise in system		05/11/18	19/11/18	14	100%	0.00
3.3.5	Improve sensor functionality		05/11/18	19/11/18	14	100%	0.00
3.4	Code Design	MB / EH	22/10/18	28/11/18	37	100%	0.00
3.4.1	I2C communication system		15/10/18	22/10/18	7	100%	0.00
3.4.2	PID implementation in code		22/10/18	05/11/18	14	100%	0.00
3.4.3	Genetic algorithm to tune PID		05/11/18	28/11/18	23	100%	0.00
3.5	Laser position sensor device	DC	23/11/18	27/11/18	4	100%	0.00
3.5.1	CAD design and 3D printing of parts		23/11/18	27/11/18	4	100%	0.00
3.5.2	Assembly of lasers and receivers with the parts		23/11/18	27/11/18	4	100%	0.00
4	**2D Simulation**	MR / YX	01/11/18	19/11/18	18	100%	0.00
4.1	Development of Transfer Function		01/11/18	19/11/18	18	100%	0.00
4.2	Control strategies for 2D		01/11/18	19/11/18	18	100%	0.00
4.2.1	Assess options and examine previous works		01/11/18	08/11/18	7	100%	0.00
4.2.2	Select control strategy and develop it		08/11/18	19/11/18	11	100%	0.00
4.3	Lead / lag compensator		08/11/18	19/11/18	11	100%	0.00
4.4	Provide a workable solution to the problem		12/11/18	19/11/18	7	100%	0.00
5	**Report**		05/11/18	29/11/18	24	100%	0.00





Code Base

```
import time
import RPi.GPIO as GPIO
import curses
import i2c
import controller
import calibrate
import PID


def range_check(value):
        if value > 100:
                value = 100
        elif value < 0:
                value = 0
        return value


screen = curses.initscr()
screen.refresh()
curses.cbreak()
screen.keypad(True)
value = 1

i2c = i2c.I2C()
pwm = controller.PWM()
v_min = v_max = 0

try:
        screen.addstr('press any key to start')
        key = 0

        while (key != 27): #Escape key
                key = screen.getch()
                voltage = i2c.getVoltage()

                if key == curses.KEY_UP:
                        value = value + 1
                elif key == curses.KEY_RIGHT:
                        value = value + 10
                elif key == curses.KEY_DOWN:
                        value = value - 1
                elif key == curses.KEY_LEFT:
                        value = value - 10
                elif key == curses.KEY_ENTER or key == 10 or key == 13:
                        value = 0
                elif key == 115: #letter s
                        cal = calibrate.Calibrate(i2c, pwm)
```

```
            v_min, v_max = cal.setup()
            screen.addstr(str(v_max))
            time.sleep(1)
        elif key == 32: #SPACEBAR
            location = (v_min + v_max) / 2
            pid = PID.PID(pwm, i2c, v_max, v_min)
            pid.position(location)

        value = range_check(value)
        pwm.DC(value)

        screen.clear()
        screen.addstr('percentage␣power␣=␣')
        screen.addstr(str(value))
        screen.addstr('␣␣␣␣␣␣␣')
        screen.addstr('sensor␣voltage␣=␣')
        screen.addstr(str(voltage))
        screen.addstr('␣␣␣␣␣␣')

        #screen.addstr(str(key))

except Exception as e:
    curses.endwin()
    pwm.cleanup()
    raise

curses.endwin()
pwm.cleanup()

# import click

# while (1):
#       a = click.getchar()
#       print(a)
#       if a == ' ':
#               break
```

```
import i2c
import time


class Calibrate(object):
        def __init__(self, i2c, pwm):
                self.i2c = i2c
                self.pwm = pwm

        def setup(self):
                v_max = self.bottom()
                v_min = self.top()

                return v_min, v_max

        def bottom(self):
                self.pwm.DC(20)
                v_min = self.average()
                self.pwm.DC(0)

                return v_min

        def top(self):
                self.pwm.DC(100)
                time.sleep(2)
                v_min = self.average()
                self.pwm.DC(0)

                return v_min

        def average(self):
                time.sleep(2)
                v_1 = self.i2c.getVoltage()
                time.sleep(0.1)
                v_2 = self.i2c.getVoltage()
                time.sleep(0.1)
                v_3 = self.i2c.getVoltage()
                v_ave = ( v_1 + v_2 + v_3 ) / 3

                max_difference = 0.02

                if ( abs(v_ave - v_1) > max_difference or abs(v_ave - v_2) > ma
                        raise ValueError('Signal␣to␣noise␣ratio␣too␣big,␣try␣blo

                return v_ave
```

```
if __name__ == '__main__':
        import i2c
        import controller
        import calibrate

        cal = Calibrate(i2c, pwm)
```

```
"""
Modified GA code from SOURCE
"""

import random
import math
import numpy as np
import csv
import time
import RPi.GPIO as GPIO
import curses
import i2c
import controller
import calibrate
import PID

MAX_TIMESTEPS = 150
POPULATION_SIZE = 50 # 100 (my values are from a paper on GAs for magnetic levit
                     # http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.
MUTATION_PROBABILITY = 0.33 # 0.1
CROSSOVER_RATE = 0.9 # 0.7
MAX_RUNS = 100 # generations
FITNESS_THRESHOLD = 5
MAX_GAIN_VALUE = 1
LINE_SMOOTHNESS = .1

class Chromosome:
    def __init__(self, kp, ki, kd):
        self.kp = kp
        self.ki = ki
        self.kd = kd


"""
1 [Start] Generate random population of n chromosomes (suitable solutions for th
Creates a random genome
"""
def generate_initial_population():
    print("Generating initial population...\n")
    random.seed()
    population = []
    for i in range(POPULATION_SIZE):
        population.append(i)
        # create a random chromosome with a random gain value
        population[i] = Chromosome(random.random() * MAX_GAIN_VALUE * 1000, rand
    return population

"""
2 [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
```

```
returns the fitness value according to the fitness function
Fitness is how long the ball remains inbetween the top and bottom without touchi
"""
def _fitness(samples): # **MAKE THIS WORK...**
    # 1. get top and bottom voltage values (as we already do)
    # 2. measure voltage at every time interval (as we already do)
    # 3. while measured voltage != (top or bottom voltages): keep going, don't c
    # 4. when measured voltage == (top of bottom voltages): fitness = number of
    #                                                        this might take a wh
    abs_errors = np.absolute(samples)
    sum_errors = np.sum(abs_errors)
    fitness = 1 / sum_errors

    return fitness

"""
Run simulation for a specific chromosome c.
Returns the fitness function value of the simulation
"""
def run_simulation_for_chromosome(population, chromosome): # **MAKE THIS WORK AN

    fitness = 0
    pid = PID.PID(pwm, i2c, v_max, v_min, position, population[chromosome].kp, p
    print("\nkp␣=␣{}\nki␣=␣{}\nkd␣=␣{}\n".format(population[chromosome].kp, popu
    errors = pid.position() # performs PID control and returns location data for
    pwm.DC(0)

    fitness = _fitness(errors)

    print('fitness␣is␣{}'.format(fitness))
    # logging
    # length = np.transpose(np.linspace(0,999,1000))
    # location = np.transpose(location)
    # myData = [length, location]
    # date = datetime.datetime.now().strftime("%H-%M-%S-%B-%d-%Y")
    # filename = './logs/PID-Response-' + chromosome + date +'.csv' # this won't
    # myFile = open(filename, 'w')
    # with myFile:
    #     writer = csv.writer(myFile)
    #     writer.writerows(myData)


    fitness += 1

    return fitness

"""
From Emil's code
"""
```

```
def range_check(value):
    if value > 100:
        value = 100
    elif value < 0:
        value = 0
    return value

"""
Run the simulation for the set of all chromosomes
"""
def run_simulation(population, generation):
    fitness_values = np.zeros(POPULATION_SIZE)
    for chromosome in range(POPULATION_SIZE):
        print("Generation: {}".format(generation))
        fitness_values[chromosome] = run_simulation_for_chromosome(population, c
        time.sleep(0.5)

    return fitness_values

"""
Pick two parents according to probability represented by normalized fitness valu
3a[Selection] Select two parent chromosomes from a population according to their
"""
def selection(fitness_values):
    # normalize the list so we have probabilities to pick parents
    fitness_values = normListSumTo(fitness_values, 1)
    parents = []
    random.seed()
    parent1_probability = random.random()
    parent2_probability = random.random()

    sum = 0
    for i in range(POPULATION_SIZE):
        if len(parents) == 2:
            break
        next_sum = sum + fitness_values[i]
        if parent1_probability <= next_sum and parent1_probability >= sum:
            parents.append(i)
        if parent2_probability <= next_sum and parent2_probability >= sum:
            parents.append(i)
        sum = next_sum
    return parents

def normListSumTo(L, sumTo=1):
    '''normalize values of a list to make it sum = sumTo'''

    total = sum(L)
    return [ x/(total*1.0)*total for x in L]
```

```
"""
3b[Crossover] With a crossover probability cross over the parents to form a new
If no crossover was performed, offspring is an exact copy of parents.
"""
def crossover(population, parents):
    random.seed()

    # if we dont crossover, offspring is a copy of parents
    if random.random() > CROSSOVER_RATE:
        return population[parents[0]]
    else:
        # random combination crossover
        number = random.random()
        if number < .25:
            return Chromosome(population[parents[1]].kp,population[parents[1]].k
        elif number < .5:
            return Chromosome(population[parents[0]].kp,population[parents[1]].k
        elif number < .75:
            return Chromosome(population[parents[0]].kp,population[parents[0]].k
        else:
            return Chromosome(population[parents[0]].kp,population[parents[0]].k

"""
3c[Mutation] With a mutation probability mutate new offspring at each locus (pos
"""
def mutation(chromosome):
    random.seed()

    if random.random() < MUTATION_PROBABILITY / 3:
        #very small real valued mutation
        chromosome.kp = chromosome.kp + random.random()/MAX_GAIN_VALUE
        if chromosome.kp < 0:
            chromosome.kp = random.random() * MAX_GAIN_VALUE

    elif random.random() < MUTATION_PROBABILITY * 2/3:
        chromosome.ki = chromosome.ki + random.random()/MAX_GAIN_VALUE
        if chromosome.ki < 0:
            chromosome.ki = random.random() * MAX_GAIN_VALUE

    elif random.random() < MUTATION_PROBABILITY:
        chromosome.kd = chromosome.kd + random.random()/MAX_GAIN_VALUE
        if chromosome.kd < 0:
            chromosome.kd = random.random() * MAX_GAIN_VALUE

    return chromosome

"""
3 [New population] Create a new population by repeating following steps until th
"""
```

```
def generate_new_population(fitness_values, previous_population, generation):
    new_population = []
    print("\nGenerating new population...\n")
    with open(filename, "w") as my_file:
        writer = csv.writer(my_file)
        for i in range(POPULATION_SIZE-1): # saves one space for elitist selecti
            new_population.append(i)
            # selection
            parents = selection(fitness_values)

            # crossover
            chromosome = crossover(population, parents)

            # mutation
            chromosome = mutation(chromosome)
            new_population[i] = chromosome
            # log kp, ki, kd, generation, chromosome (relative)
            kp = population[chromosome].kp
            ki = population[chromosome].ki
            kd = population[chromosome].kd
            my_data = [generation, chromosome, kp, ki, kd, 0]
            filename = './logs/GA_PID_parameters' + '.csv'
            writer.writerows(my_data)


    """
    Perform hybrid elitist selection. Carry the best chromosome over to the new
    """
    chromosome = population[np.argmax(fitness_values)]
    new_population.append(POPULATION_SIZE-1)
    new_population[POPULATION_SIZE-1] = chromosome
        # log kp, ki, kd, generation, chromosome (relative)
    kp = population[chromosome].kp
    ki = population[chromosome].ki
    kd = population[chromosome].kd
    my_data = [generation, chromosome, kp, ki, kd, 1]
    filename = './logs/GA_PID_parameters' + '.csv'
    with open(filename, "w") as my_file:
        writer = csv.writer(my_file)
        writer.writerows(my_data)

    return new_population


def file_len(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1
```

```
"""
    Main
    1 [Start] Generate random population of n chromosomes (suitable solutions fo
    2 [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
    3 [New population] Create a new population by repeating following steps unti
        3a[Selection] Select two parent chromosomes from a population according
        3b[Crossover] With a crossover probability cross over the parents to for
        3c[Mutation] With a mutation probability mutate new offspring at each lo
        3d[Accepting] Place new offspring in a new population
    4 [Replace] Use new generated population for a further run of algorithm
    5 [Test] If the end condition is satisfied, stop, and return the best soluti
    6 [Loop] Go to step 2
"""

##########################################################################################
#
#   NOTE: Fitness_values is indexed by chromosome number, so if we attempt to so
#   we will pick the wrong index when generating a new population. Which would b
#
##########################################################################################

# screen = curses.initscr()
# screen.refresh()
# curses.cbreak()
# screen.keypad(True)
# value = 1

generation = 1

i2c = i2c.I2C()
pwm = controller.PWM()
v_min = v_max = 0

try:
    # screen.addstr('press any key to start')
    # key = 0

    cal = calibrate.Calibrate(i2c, pwm)
    v_min, v_max = cal.setup()
    position = (v_min + 2*v_max)/3
    filename = './logs/GA_PID_parameters' + '.csv'
    try:
        f = file.open(filename)
        f.close()
    except FileNotFoundError:
        print('\nFile does not exist, creating new log file\n')
        header = ["generation", "chromosome_no.", "kp", "ki", "kd", "elite?"]
        with open(filename, "w") as my_file:
                writer = csv.writer(my_file)
                writer.writerows(header)
        if file_len(filename) > POPULATION_SIZE:
            with open(filename, "r") as my_file:
                reader = csv.reader(my_file)
                generation = reader[-1][0] # final chromosome's generation no. (shou
                population = []
                for line in reader:
                    kp = line[2]
                    ki = line[3]
                    kd = line[4]
                    chromosome = Chromosome(kp, ki, kd)
                    population.append(chromosome) # population is continued from csv
        else:
            population = generate_initial_population()

        fitness_values = run_simulation(population)
        time.sleep(1)
    except Exception as e:
        raise ValueError(e)


    for i in range(MAX_RUNS - generation):
        print("Generation {}".format(generation))
        population = generate_new_population(fitness_values, population, generation)
        fitness_values = run_simulation(population, generation)

        max_value = max(fitness_values)

        print("Maximum fitness of generation {} = {}".format(generation, max_value))

        # if the population sucks, DESTROY THE EARTH
#       if max_value < FITNESS_THRESHOLD:
 #          print("Population sucked so we're starting with a fresh batch lol")
  #         population = generate_initial_population()
   #        generation = 1
    #       continue

        generation += 1
```

```
import smbus
import time


class I2C(object):

        def __init__(self):
                self.address = 0x04
                self.bus = smbus.SMBus(1)
                self.value = 1

        def getVoltage(self):
                flag = 0
                while flag == 0:
                        try:
                                self.writeNumber(self.value)
                                number = self.readNumber()
                                temp = ''.join(str(x) for x in number)
                                voltage = float(temp)/ 1000
                                flag = 1
                        except:
                                pass




                return voltage

        def writeNumber(self, value):
                self.bus.write_byte(self.address, value)
        # bus.write_byte_data(address, 0, value)
                return -1

        def readNumber(self):
            #number = bus.read_byte(address)
            number = self.bus.read_i2c_block_data(self.address, 0, 4)
            #number = bus.read_byte_data(address, 1)
            return number
```

```
import datetime
import csv
import numpy as np


class PID(object):

        def __init__(self, pwm, i2c, v_max, v_min):
                self.pwm = pwm
                self.i2c = i2c
                self.v_max = v_max
                self.v_min = v_min

        def force_normaliser(self, location, G):
                '''As the attractive force depends on the distance from the magn
                The force of the electromagnet is dependent on the square of the
                '''

        #checking the reading for errors
                if location < self.v_min - 0.5:
                        raise ValueError(location, self.v_min, 'position read as
                operation_range = self.v_max - self.v_min
        #top location is v_min, bottom location is v_max (because a higher posit
                normalised_distance_from_top = (location - self.v_min) / operati
                if normalised_distance_from_top > 1.5:
                        raise ValueError(normalised_distance_from_top, 'NDFT is 
                elif normalised_distance_from_top > 1:
                        normalised_distance_from_top = 1
                force = G * normalised_distance_from_top ** 2
                self.pwm.DC(force)

                return force

        def logger(self, timestep, location, force):
                location = np.transpose(location)
                myData = [timestep, location, force]
                date = datetime.datetime.now().strftime("%H-%M-%S-%B-%d-%Y")
                filename = './logs/PID-Response-' + date +'.csv'
                myFile = open(filename, 'w')
                with myFile:
                        writer = csv.writer(myFile)
                        writer.writerows(myData)


        def position(self, position):
                target_position = position
                i = 0
                error_past = 0
                Integral = 0
```

```
location = np.zeros(1000)
G = 0
peak_limit = 100
force = np.zeros(1000)
while i < 1000:
        location[i] = self.i2c.getVoltage()
        error = (location[i] - target_position)/ target_position

        KP = 600
        KI = 0.001
        KD = 10

        V = error - error_past
        D = KD * V

        P = error * KP
        if G != peak_limit:
                Integral += error
                I = Integral * KI

        G = P + I + D

        if G > peak_limit:
                G = peak_limit
        elif G < 0:
                G = 0

        force[i] = self.force_normaliser(location[i], G)

        i +=1
        error  = error_past
timestep = np.transpose(np.linspace(0,999,1000))
self.logger(timestep, location, force)
```

Curve fitting for 3D magnetic simulations

```
%This MATLAB code is written for fitting the curve of current against the
%displacement of the ball in order to find the transfer function

%Electromagnetic Levitation Group Project, Yidan Xue, 9 Oct 2018

clc; clear all;

filename = 'magnetic␣simulation␣3D.xlsx';
sheet = 1;
xlRange = 'A2:B10';
data = xlsread(filename,sheet,xlRange);

p = polyfit(data(:,1),data(:,2),2);

x1 = linspace(10,50);
y1 = polyval(p,x1);


figure
set(gcf,'color','w');
plot(data(:,1),data(:,2),'bo','linewidth',2);hold on
plot(x1,y1,'-r','linewidth',2);hold on
legend('Test␣Points','Fitting␣Curve');
title('Plot␣of␣Current␣against␣Displacement','fontsize',14);
xlabel('Distance␣between␣the␣centre␣of␣ball␣and␣electromagnetic␣(mm)','fontsize'
ylabel ('Current␣(A)','fontsize',14);
grid on;

% kc = zeros(1,9);
% kc(:) = 0.2783.*data(:,1)*10^(-6)./data(:,2);
%
% q = polyfit(data(:,1),kc(:),3);
%
% x2 = linspace(10,50);
% y2 = polyval(q,x1);
%
% figure
% plot(data(:,1),kc(:),'bo');hold on
% plot(x2,y2,'-r','linewidth',2);hold on
% legend('Test Points','Fitting Curve');
% title('Plot of Coil Constant against Displacement','fontsize',14);
% xlabel('Distance between the centre of ball and electromagnetic (mm)','fontsiz
% ylabel ('Coil constant','fontsize',14);
% grid on;
```