UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

"MODELLAZIONE E CONTROLLO DI SISTEMI MAGLEV ELLITTICI"

Relatore: Prof. Damiano Varagnolo
Università degli Studi di Padova

Laureando: Francesco De Marchi

Correlatore: PhD candidate Hans Engmark
Norges Teknisk-Naturvitenskapelige Universitet

ANNO ACCADEMICO 2021 – 2022

Data di laurea 14/11/2022

1 Abstract

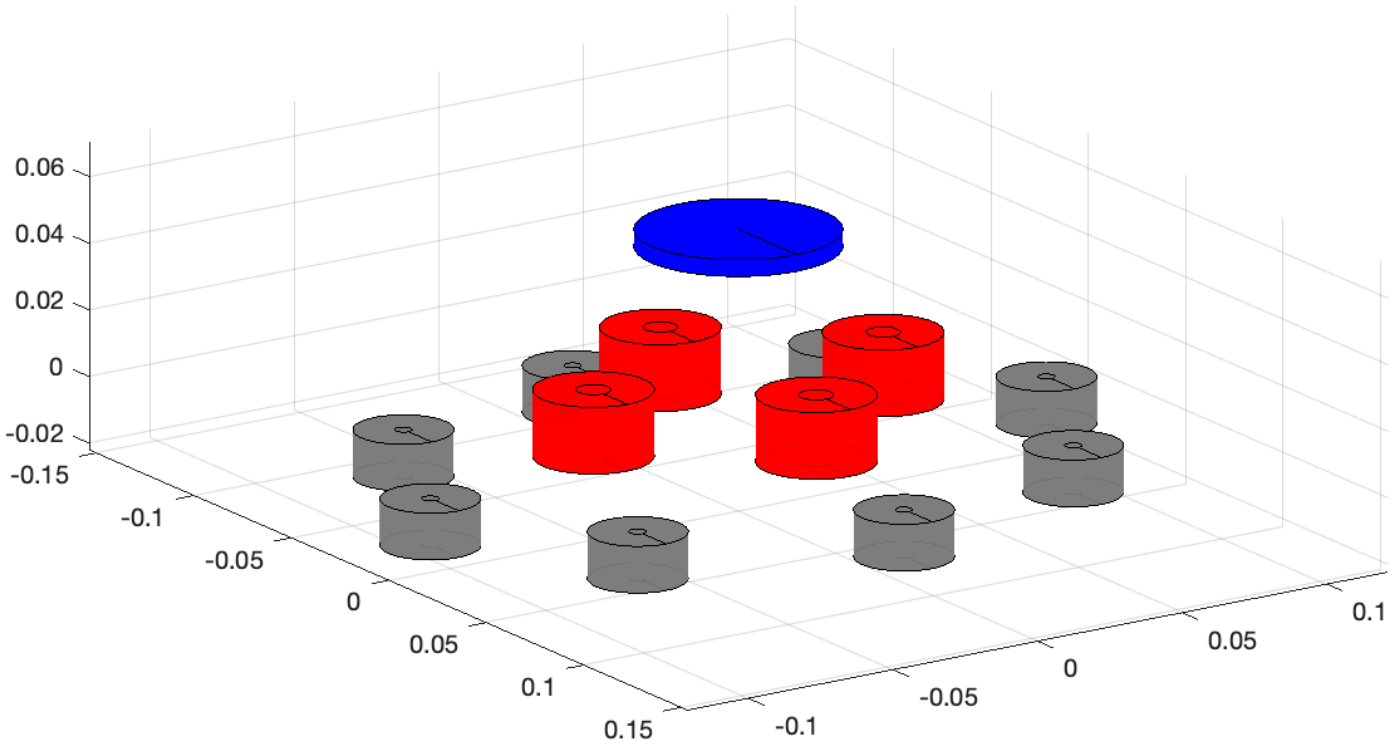
This thesis is part of a larger project in collaboration with the Norwegian University of Science and Technology, NTNU, and it has the objective of designing and building a maglev platform that is cheap, relatively easy to assemble, and reprogrammable. The intuition is that this platform may be used by students taking control systems subjects, that could then build their own system to experiment with, and develop a better understanding of the theoretical concepts behind control.

The first steps of this project were taken by a series of NTNU students in their bachelor thesis, *"Magnetic levitation systems: design, prototyping and testing of a digital PID-controller"* [3]. This first work built a platform that used permanent magnets and actively controlled solenoids (using PID control) to magnetically levitate a magnet. It also developed a mathematical framework in Matlab to simulate the system. This thesis reports part of a new round of development of the system, performed in collaboration with two other students from University of Padova, Alberto Morselli and Andrea Nicetto. This round overhauls the electrical circuits of the system, improves the control algorithms, and generalizes the results to other mechanical designs.

This thesis focuses in particular on this generalization: we attempt to modify the system design to simulate the magnetic field the magnets would generate if they were placed in an elliptical arrangement instead of the original circular arrangement. This change in the disposition of the magnets is indeed deemed as allowing to implement more sophisticated (and interesting for the users) control strategies for the levitating magnet. This adaptation requires the accomplishment of several tasks: first, altering the ODEs to suit the new design; second, changing the positions of the sensors for the magnetic field as they may not be optimal anymore; third, finding which new limitations in the ability to control the levitating magnet are given by physical components such as actuators and sensors, since they must have a limited cost for the system to be affordable.

2 Abstract in lingua Italiana

Questa tesi è parte di un progetto più grande svolto in collaborazione con l'Università Norvegese di Scienze e Tecnologie, NTNU, che come obbiettivo ha l'idealizzazione di una piattaforma per la levitazione magnetica che sia economica e relativamente facile da assemblare. Gli studenti di teoria dei sistemi di controllo avrebbero dunque l'opportunità di replicare questa piattaforma per sperimentare i concetti teorici che hanno studiato e svilupparne una miglior comprensione.
I primi passi del progetto sono stati compiuti da un gruppo di studenti del NTNU nella loro tesi triennale, *"Magnetic levitation systems: design, prototyping and testing of a digital PID-controller"* [3]. Essi costruirono una piattaforma che sfrutta dei magneti permanenti per generare un campo magnetico stabile, dei solenoidi per alterare questo campo magnetico nel modo necessario per mantenere in equilibrio un disco magnetico, e un controllore PID per gestire questi solenoidi. Inoltre svilupparono anche un simulatore del sistema in MATLAB. In questa tesi è riportata parte della seconda iterazione di questo progetto, svolta in collaborazione con altri due studenti dell'Università degli Studi di Padova, Alberto Morselli e Andrea Nicetto. In questa seconda fase del progetto è stata compiuta una ri-pianificazione dei circuiti elettrici, lo sviluppo di nuovi algoritmi di controllo e uno studio sulla generalizzazione del posizionamento dei magneti. Questa tesi in particolare è incentrata su questa ultima parte: è stato studiato in simulazione il campo magnetico che i magneti genererebbero se fossero disposti in una conformazione ellittica anzichè circolare, come nel sistema fisico. Questa ricerca è stata ritenuta interessante in quanto potrebbe portare allo sviluppo di strategie di controllo più sofisticate (e interessanti per l'utilizzatore del sistema) per la levitazione dell'oggetto levitante. Queste modifiche richiedono l'analisi di diversi fattori: primo, come variano le equazioni differenziali che descrivono il sistema; secondo, come dovrebbe cambiare la posizione dei sensori per mantenere ottimale la misurazione del campo magnetico; terzo, scoprire quali sono i nuovi limiti dell'abilità di controllare l'oggetto levitante dettati dalle componenti fisiche del sistema, come attuatori e sensori, considerando che questi dovrebbero avere un costo limitato per rispettare gli obbiettivi del progetto.


0.06
0.04
0.02
0
-0.02
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
-0.1
-0.05
0
0.05
0.1

Contents

3 Introduction

A magnetic levitation (abbreviated: MagLev) platform is a system that uses magnetic fields in order to lift and control an object. This can be achieved in several ways, usually with a combination of permanent magnets and electromagnets, as in the case analysed in this thesis, or diamagnets, which are magnets that when subjected to another external magnetic field exhibit a weak magnetic field antiparallel to the first one, or superconductors, which are particular diamagnets that can create a strong magnetic field.
The applications of magnetic levitation are wide-ranging, from microrobotics to medicine, but the most famous use of MagLev technology is arguably in the field of transportation - the word Maglev itself is in fact usually associated with magnetically levitating trains.

This project focused on building a MagLev platform started in 2021 and was originally worked on by four students of the Norwegian University of Science and Technology, while this second iteration is the result of the efforts of three students, including the author of this thesis, of the Università degli Studi di Padova, who had the objective of improving both the hardware and the software components of the project.
The aim of the project was to design a system that can be an helpful tool for other control theory students, since it's application requires the understanding several fundamental concepts such as the linearization of non-linear differential equations, the transfer function, the step response and feedback loops.

The physical system is composed of a platform that holds eight magnets and four solenoids, placed in a circular arrangement on two different levels, and the electronics for the control of the system, allocated below the magnets. The specifics of all the components used to build the platform can be found in the thesis on the first iteration of the project.
The MATLAB model of the system can be used to calculate the position of the equilibrium and the area of controllability, to simulate the efficiency of a controller and to map the magnetic field and the forces exerted on the levitating disc.

This thesis will focus on some of the modifications applied to the MatLab model that was created by the first group of students and then overhauled by the second group, in particular on the scripts that evaluate what changes in the magnetic field generated by an elliptical arrangement of the magnets. Once the magnetic field has been calculated it is then possible to get an estimation of the area where the levitating disc is controllable by analyzing the force that can be applied on it by the magnets. A second, more accurate, approach to the study of this area has been explored that requires the calculation of the condition number, but while the results acquired seem compatible with those obtainable from the first approach, they remain untested on the physical system. The version of the simulator used for this thesis can be found at this link

4 Physical Background

The levitation of the magnetic disc is caused by its interaction with the magnetic fields generated by the solenoids and the permanent magnets. By altering the magnetic field of the solenoids it's also possible to control the trajectory of the disc, at least while it is within a certain region containing the equilibrium of the system. In order to be able to calculate the intensity of the magnetic field and consequently the magnetic force affecting the disc, the neodymium magnets and the levitating disc were modelled as if they were solenoids in a way described in Chapter 6. In this chapter it will be briefly explained how it's possible to calculate the magnetic force a solenoid can generate at a certain position
Solenoids are a type of magnets called electromagnets, named so for the way their magnetic field is generated by the current flowing through them. It is possible to obtain the intensity of the magnetic field generated by the current flowing in a circuit at a point P by using Biot-Savart's law, described by the following equation:

$$\vec{B} = \frac{\mu}{4\pi} \int_{\gamma} \frac{Id\vec{s}}{r^2} \vec{u_t} \times \vec{u_r} \tag{1}$$

where μ is the magnetic permeability of free space, γ is the circuit along which the current I flows with direction ds , $\vec{u_t}$ is the versor tangent to the circuit, $\vec{u_r}$ is the versor pointing toward the point P and finally r is the distance from the current element $Id\tilde{s}$ that point.

This law can thus be used to calculate the magnetic field generated by the current flowing through the solenoids at the position of the levitating disc, to calculate then the force exercised by the magnetic field on the disc, it is possible to use the second law of Laplace:

$$\vec{F} = \int_{\gamma} id\vec{s} \times \vec{B} \tag{2}$$

However, the integrals required to solve Biot-Savart's law and the second law of Laplace cannot in general be solved in this form but they can be solved through a process presented by Gonzàlez and Càrdenas (2021) [2]. The resulting equations, in cylindrical coordinates, are:

$$B(\rho, \phi, z) = \frac{\mu_0 I \rho'}{4\pi} (\vec{B_x} + \vec{B_y} \vec{B_z}) \tag{3}$$

$$\vec{B_x} = \vec{e_x}(z - z') \int_{\phi_1}^{\phi_2} \frac{cos(\phi')}{[\rho'^2 + \rho^2 + (z - z')^2 - 2\rho'\rho cos(\phi' - \phi)]^{\frac{3}{2}}} d\phi' \tag{4}$$

$$\vec{B_y} = \vec{e_y}(z - z') \int_{\phi_1}^{\phi_2} \frac{sin(\phi')}{[\rho'^2 + \rho^2 + (z - z')^2 - 2\rho'\rho cos(\phi' - \phi)]^{\frac{3}{2}}} d\phi' \tag{5}$$

$$\vec{B_z} = \vec{e_z} \int_{\phi_1}^{\phi_2} \frac{\rho' - \rho cos(\phi' - \phi)}{[\rho'^2 + \rho^2 + (z - z')^2 - 2\rho'\rho cos(\phi' - \phi)]^{\frac{3}{2}}} d\phi' \tag{6}$$

The last concept needed to understand the physical background of the system is the Hall effect, since the sensors used to measure the magnetic field are Hall probes, or Hall effect sensors.
Such an effect explains how the presence of a magnetic field affects the motion of a charged particle inside a conductor exposed to an electric field. The formula to determine the effect is:

$$B = \frac{ned}{I} V_H \tag{7}$$

Where n is the volumetric density of the particles, e is the elementary charge, d is the length of the equipotential side of the conductor through which the particles are flowing, I is the current intensity and V_H is the potential applied to the conductor.
If I is constant and is known, then $\frac{ned}{I}$ is constant and known as well, so by exposing the Hall sensor to a known magnetic field it is possible to quantify V_H. By measuring V_H it is then possible to obtain the intensity of the component of the magnetic field perpendicular to the circuit in the sensor, this means that at least three Hall sensor are necessary to measure all three components of $\vec{B}$.

5 Modelling of the system

To model the system effectively, some simplifications were introduced: as stated in the previous chapter, all the magnets were treated as solenoids for the calculation of the magnetic forces, these solenoids were then approximated as n_r concentric stacks of n_h coils each. Moreover, the coils were considered as regular polygons of n sides, in this way it was possible to obtain the force and torque of the levitating disc by calculating the forces exercised by each segment of each coil using equation (2)

The system can be modelled by using the Newton-Euler equations of motion, as written in "Model Description: Magnetic Levitating System", by Hans Engmark.

$$\begin{bmatrix} \sum_{i=1}^{n} F_i \\ \sum_{i=1}^{n} \tau_i \end{bmatrix} = \begin{bmatrix} mI_3 & 0 \\ 0 & \mathscr{I} \end{bmatrix} \begin{bmatrix} a \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \times \mathscr{I}\omega \end{bmatrix} \tag{8}$$

Where F_i and τ_i are the forces and torques exercised by each segment of the coils, m is the mass of the levitating disc, I is the Identity matrix, $\mathscr{I}$ is the matrix of inertia, and a and α are the linear and angular accelerations respectively.

As state of the system, it was chosen to use:

$$\eta = \begin{bmatrix} x & y & z & \psi & \theta & \phi & \dot{x} & \dot{y} & \dot{z} & \dot{\psi} & \dot{\theta} & \dot{\phi} \end{bmatrix}^T \tag{9}$$

The first six variables are the space coordinates and the angles of the disc with the axes, the second six variables are the derivatives of the first six.
The equation of each component can be obtained by inverting the Newton-Euler's laws of motion (8), the result is the following model, better described in Engmark's document:

$$\begin{cases} \dot{\eta} = \begin{bmatrix} O_{6x6} & I_6 \\ O_{6x6} & O_{6x6} \end{bmatrix} \eta + \begin{bmatrix} O_{6x6} \\ I_6 \end{bmatrix} \sigma(\eta, u) \\ \\ y = \begin{bmatrix} B_x(\eta, u) \\ B_y(\eta, u) \\ B_z(\eta, u) \end{bmatrix} \end{cases} \tag{10}$$

$\sigma(\eta, u)$ being the six dimensional column vector:

$$\sigma(\eta, u) = \begin{bmatrix} mI_3 & O_{3x3} \\ O_{3x3} & \mathscr{I} \end{bmatrix}^{-1} \begin{bmatrix} F_x(\eta, u) \\ F_y(\eta, u) \\ F_z(\eta, u) \\ \tau_x(\eta, u) \\ \tau_y(\eta, u) \\ \tau_z(\eta, u) \end{bmatrix} + \begin{bmatrix} O_{2x1} \\ -g \\ O_{3x1} \end{bmatrix} \tag{11}$$

The output of the model is $\vec{B}$, with each component being measured by one of the three Hall effect sensors for the reasons described in the last chapter. In order to get the most precise readings as possible from the sensors, they where placed in the origin of the system, directly below the equilibrium, where the variation of the magnetic field in case of a displacement is the highest and the relative error is then the lowest.

The equations of the system can describe a general disposition of both the solenoids and the permanent magnets, since none of the terms were simplified due to any symmetry of the original circular arrangement, and thus it wasn't necessary to alter them in any way to adapt them to an elliptical arrangement of the magnets.
The point of equilibrium of the system, as explained in "Magnetic Levitation System: Design, Prototyping And Testing Of A Digital PID-Controller" (2022) [3], is in the centre of the system, where the lateral components of the magnetic forces exercised by the magnets cancel each other out and the vertical components are canceled out by the gravitational force. However, altering the position of the permanent magnets from the original circular position would alter the height of the point of equilibrium, since by increasing or reducing their distance from the original point of equilibrium would result in a different intensity of the force applied at that point, causing it to move up or down.
A deeper analysis of the characteristics of the equilibrium, as well as other aspects of the system that can be altered by changing the disposition of the magnets, can be found in the next chapter.

6 Matlab Implementation

6.1 Description of the simulator

In order to study the characteristics of the system, such as the value of the fictitious current flowing in the permanent magnets, the position of the points of equilibrium and the limits of the controllable area, a simulator of the model was implemented in MATLAB.
This chapter describes the scripts that needed to be adapted to represent a system with an elliptical disposition of the magnets, but all the scripts used can be found at (todo: put link to github).

A standard ellipse is described by the equation:

$$y = \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{12}$$

where a^2 and b^2 are the lengths of the x and y axes respectively. It was arbitrarily chosen to consider the x axis as unitary and to study the system with the y axis as the only variable axis. This allowed for the ellipticity of the system to be describable using only one parameter instead of two, and it was then possible to describe the ellipse using only its eccentricity, which is defined as:

$$e = \sqrt{1 - \frac{a^2}{b^2}} \tag{13}$$

Since a was considered as unitary, the length of the y axis can be derived form the eccentricity by inverting this equation.
The eccentricity is an input of the scripts necessary to model the system and the most important of them is maglevSystem.m, from which the following snippet is from.

```
function obj = maglevSystem(x0, params, approximationType,
    eccentricity)
       mu0 = 4*pi*1e-7;
       obj.approximationType = approximationType;
       bsqr = 1 / (1 - eccentricity^2); % multiplier for the length
           of the y axis of the ellipse

       %% Solenoids
       Xs = params.solenoids.R*cos(linspace(0,2*pi,params.solenoids.N
           +1));
       Ys = params.solenoids.R*bsqr*sin(linspace(0,2*pi,params.
           solenoids.N+1));
       Zs = zeros(size(Xs));

       obj.SOLENOIDS = [solenoid(1,1,1,1,1,1,zeros(12,1),0,0,0,'r')];
       for i = 1:params.solenoids.N
           obj.SOLENOIDS(i) = solenoid(params.solenoids.ri,params.
               solenoids.ro,params.solenoids.h, ...
```

```
                                    params.solenoids.nr,params.
                                        solenoids.nh,params.solenoids
                                        .nl, ...
                                    [Xs(i),Ys(i),Zs(i)-params.
                                        solenoids.h/2 + params.
                                        solenoids.zs,
                                        0,0,0,0,0,0,0,0,0]', ...
                                    1,100*mu0,approximationType,'r');
        end

        %% Magnets
        Xpm = params.magnets.R*cos(linspace(0,2*pi,params.magnets.N+1)
            +params.magnets.offset);
        Ypm = params.magnets.R*bsqr*sin(linspace(0,2*pi,params.magnets
            .N+1)+params.magnets.offset);
        Zpm = zeros(size(Xpm));

        obj.MAGNETS = [solenoid(1,1,1,1,1,1,zeros(12,1),0,0,0,'r')];
        for i = 1:params.magnets.N
            obj.MAGNETS(i) = solenoid(params.magnets.ri,params.magnets
                .ro,params.magnets.h, ...
                                    params.magnets.nr,params.magnets.nh
                                        ,params.magnets.nl, ...
                                    [Xpm(i),Ypm(i),Zpm(i)-params.
                                        magnets.h
                                        /2,0,0,0,0,0,0,0,0,0]', ...
                                    params.magnets.I,mu0,
                                        approximationType,0.5*ones
                                        (1,3)); %==gray
        end

        %% Floating magnet
        obj.LEVITATINGMAGNET = solenoid(params.levitatingmagnet.ri,
            params.levitatingmagnet.ro,params.levitatingmagnet.h, ...
                                    params.levitatingmagnet.nr,params
                                        .levitatingmagnet.nh,params.
                                        levitatingmagnet.nl, ...
                                    x0,params.levitatingmagnet.I,mu0,
                                        approximationType,'b');
        obj.m = params.levitatingmagnet.m;

        %% Sensors
        obj.xSens = params.sensor.x;
        obj.ySens = params.sensor.y;
        obj.zSens = params.sensor.z;
    end
```

The only part that of the script that was necessary to alter was the constructor of the class maglevSystem, here presented: the eccentricity was included in

the parameters of the constructor and is used to calculate the variable *bsqr* which is then used as multiplicative factor for the positioning of both the solenoids and the magnets along the y axis. It is thus possible to simulate the system with an elliptical configurations of just the magnets or just the solenoids, or for example to align the ellipses along the x axis instead, by just removing the *bsqr* factor or changing which array of coordinates is multiplied by it.

6.2 Force Plotter

Another script that had to be slightly adapted is force_plotter.m, renamed force_plotter_axis.m. The original script maps the forces exercised on the levitating disc in n^3 points of a cube centered around a point slightly lower than the equilibrium. The new version of the script maps the forces only along the XZ and YZ planes instead, to give a more clear representation of the differences between the forces felt by the disc along the planes containing the axes of the ellipse, and plots a region of space that is proportional to the length of the major axis of the ellipse.

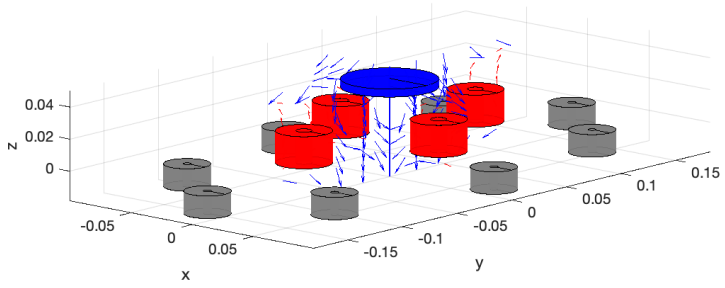
```
%{
    maps the force felt by the levitating magnet in a 3d cube of points
%}

clear; close all;
addpath('../maglevFunctions');
load('params.mat');
load('results.mat');

approximationType = input("approxType [0/1]> ");
eccentricity = input("Eccentricity [0:1]> ");
eccentricity = max (eccentricity, 0); eccentricity = min (eccentricity,
    1);

%% Searching parameters
steps = 9; % odd number for planes along the axis
L = .10;
bsqr = 1 / (1 - eccentricity^2);

if(approximationType == 0)
    eq = results.zeq.zeq_fst;
    params.magnets.I = results.neo_vs_neo.curr_fst;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_fst;
else
    eq = results.zeq.zeq_acc;
    params.magnets.I = results.neo_vs_neo.curr_acc;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_acc;
end
```

```
%% Derived parameters
Ptx = linspace(-L/2, L/2, steps);
Pty = linspace(-L/2*bsqr, L/2*bsqr, steps); % assuming the ellipse's
    major axis is y
Zs = linspace(eq-L*9/20, eq+L*1/10, steps);
Fzs = zeros(length(Ptx), length(Pty), length(Zs),3); %3D matrix of a 3-
    page vals

x0 = zeros(12,1); x0(3) = eq;
u = [0; 0; 0; 0]; % x+, y+, x-, y-
sys = maglevSystem(x0, params, approximationType, eccentricity);

% YZ Plane
i = ceil(length(Ptx)/2); % x = 0
for j = 1:length(Pty) %y
    for k = 1:length(Zs) %z
        x0(1) = Ptx(i); x0(2) = Pty(j); x0(3) = Zs(k); %  YZ plane
        temp = sys.f(x0, u);
        % normalize to make a good plot
        vec = temp(7:9);
        Fzs(j,i,k,:) = vec./(norm(vec)); %x,y,z forces
    end
end

%% XZ Plane
j = ceil(length(Pty)/2); % y = 0
for i = 1:length(Ptx) %x
    for k = 1:length(Zs) %z
        x0(1) = Ptx(i); x0(2) = Pty(j); x0(3) = Zs(k); %  XZ plane
        temp = sys.f(x0, u);
        % normalize to make a good plot
        vec = temp(7:9);
        Fzs(j,i,k,:) = vec./(norm(vec)); %x,y,z forces
    end
end


%% Plotter
figure(1);
[X,Y,Z] = meshgrid(Ptx, Pty, Zs);
[U,V,W] = deal(Fzs(:,:,:,1), Fzs(:,:,:,2), Fzs(:,:,:,3));

idx = Fzs(:,:,:,3) >= 0;
quiver3(X(idx),Y(idx),Z(idx), U(idx),V(idx),W(idx), .6, 'r'); hold on;
quiver3(X(~idx),Y(~idx),Z(~idx), U(~idx),V(~idx),W(~idx), .6, 'b'); hold
    on;
grid on; axis equal; view([45,15]);
xlabel('x'); ylabel('y'); zlabel('z');

load params;
```

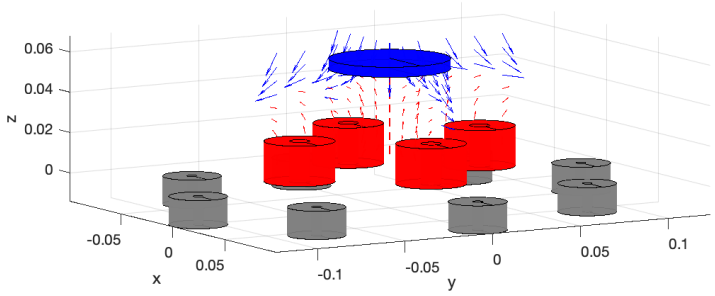
```
%params.solenoids.ri = 0;
%params.solenoids.ro = 0; % solenoids are covering the forces, better to
    not plot them
x0 = zeros(12,1); x0(3) = eq;
sys = maglevSystem(x0, params, approximationType, eccentricity);
draw(sys, 'fancy'); hold off;
```

It is possible to simulate the forces applied on the disc by any combination of currents flowing in the solenoids by modifying the value of the variable u in the section *Derived parameters*, here set to 0 for all the four solenoids.

Figure 1: Some examples of the graphs obtainable from this script:



(a) With an eccentricity of 0.7 the magnets are too far to garantee stability



(b) With an eccentricity of 0.47 it's possible to stabilize the disc

6.3 Equilibrium Finder

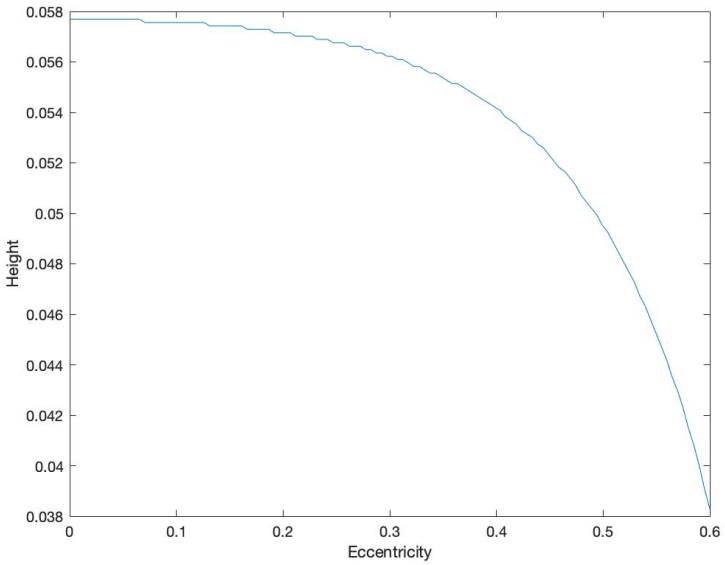
The script eq_finder can be used to observe the relation between the eccentricity of the ellipse of the magnets and the height of the point of equilibrium. Naturally the positions of the solenoids don't affect the results given by this script, since the equilibrium is calculated for an autonomous system it depends only on the position of the permanent magnets.

```
%{
    calculates the height of the equilibrium in (x=0,y=0)
    for different eccentricities of the ellipse

    an ellipses is described by the equation
    x^2/a^2 + y^2/b^2 = 1
    sqrt(1-a^2/b^2) is the eccentricity of the ellipse
%}

clear; close all;
addpath('../maglevFunctions');
load('params.mat');
load('results.mat')

approximationType = input("approxType [0/1]> ");

%% Searching parameters
min_height = .03;
max_height = .07;
height_steps = 300;
min_eccentricity = 0;
max_eccentricity = .6;
eccentricity_steps = max_eccentricity * 200; % plotting every .005

if(approximationType == 0)
    eq = results.zeq.zeq_fst;
    params.magnets.I = results.neo_vs_neo.curr_fst;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_fst;
else
    eq = results.zeq.zeq_acc;
    params.magnets.I = results.neo_vs_neo.curr_acc;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_acc;
end

x0 = zeros(12,1); x0(3) = eq;

%% Finding the height of the equilibrium for each eccentricity
Zs = linspace(min_height, max_height, height_steps); % heights at which
     the force will be calculated
Es = linspace(min_eccentricity, max_eccentricity, eccentricity_steps); %
     eccentricities evaluated
Bs = zeros(size(Es)); % will contain the length of the Y axis of the
```

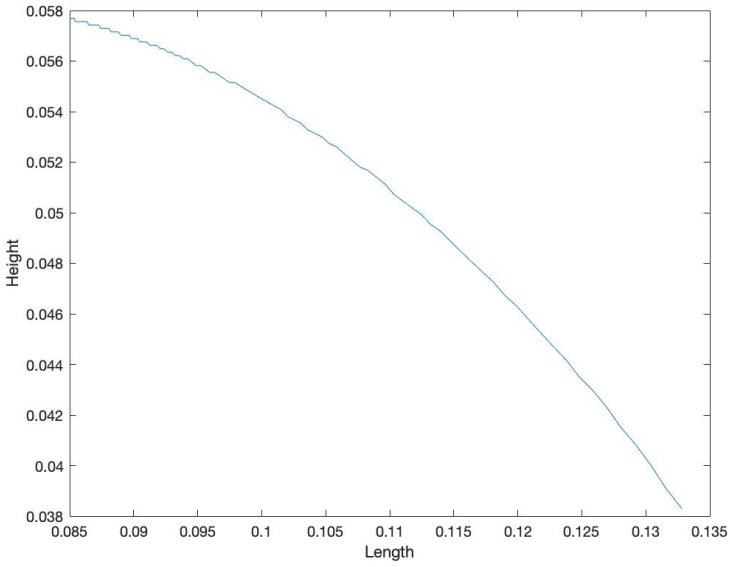
```
    ellipse
Fzs = zeros(size(Zs)); % vector that will contain the force at each
    height
Heights = zeros(size(Es)); % vector with the results

h = waitbar(0);
for i = 1:length(Es)
    Bs(i) = params.magnets.R*(1/(1-Es(i)^2));
    sys = maglevSystem(x0, params, approximationType, Es(i)); % modelling
         a system of eccentricity Es(i)
    for k = 1:length(Zs)
        temp = sys.f([0,0,Zs(k),zeros(1,9)]',zeros(params.solenoids.N,1))
            ;
        Fzs(k) = temp(9);
    end
    [~,idx]=min(abs(Fzs));
    Fzs = zeros(size(Zs)); % resetting Fzs for precaution
    Heights(i) = Zs(idx); % saving the height of the equilibrium
    waitbar(i/length(Es))
end
close(h);

%% Plotting
figure('Name', 'Height of the equilibrium / Eccentricity');
grid minor; axis equal;
plot(Es, Heights)
xlabel('Eccentricity'); ylabel('Height');

figure('Name', 'Height of the equilibrium / Length of the Y axis');
grid minor; axis equal;
plot(Bs, Heights)
xlabel('Length'); ylabel('Height');
```

For the sake of completeness, the script plots two graphs: one to compare the height of the equilibrium and with eccentricity, and one to compare it to the length of the y axis of the ellipse. While the first graph is more related with the results of the other scripts, which are all dependable on the eccentricity, the second graph shows more clearly the correlation between the height of the equilibrium and the dimensions of the ellipse.
These graphs are shown in Figure 2a and 2b.

Figure 2: Height of the equilibrium



(a) Height of the equilibrium in respect to the eccentricity



(b) Height of the equilibrium in respect to the length of the ellipse

6.4 Controllability

The area where the disc is controllable has been proven to be convoluted to calculate, but it's possible to obtain an approximation of it.
The script eccentricity_controllability.m is a script that operates in a similar way to that of solenoids_controlling_power.m: for a give set of eccentricities, it calculates the height of the equilibrium, then proceeds to map the x and y components of the linear acceleration along the respective axes on the XY plane containing the equilibrium the same process is then repeated for the z axis. If these steps are computed for the highest possible intensity of the input of the system, then the cuboid limited by the points where the acceleration is null is an approximation of the actual controllable area, more specifically it circumscribes it.

```
%{
    measures the volume of the controllable area*
    for different eccentricities of the system (both magnets and
        solenoids)

    use the paramiters / figures for a single OR varying eccentricity, do
    not use both

    *the exact controllable area is difficult to measure, so the
    smallest circumbscribing rectangoloid is calculated instead
%}

clear; close all;
addpath('../maglevFunctions');
load('params.mat');
load('results.mat');

approximationType = input("approxType [0/1]> ");

%% Searching parameters
steps = 512;
L = .10;
% parameters for a varying eccentricity
min_eccentricity = 0; max_eccentricity = .5;
eccentricity_steps = max_eccentricity * 200; % plotting every .005
% parameters for a single eccentricity to check if the program
% works (faster to compare with solenoids_controlling_power
% and z_graph)0
%min_eccentricity = .25; max_eccentricity = min_eccentricity;
%eccentricity_steps = 1;

% load correct parameters
if(approximationType == 0)
    eq = results.zeq.zeq_fst;
    params.magnets.I = results.neo_vs_neo.curr_fst;
```

```
    params.levitatingmagnet.I = results.neo_vs_lev.curr_fst;
else
    eq = results.zeq.zeq_acc;
    params.magnets.I = results.neo_vs_neo.curr_acc;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_acc;
end

%% Derived parameters
% Doubles the space for two halfs respectively for positive error
% (solenoids = +-) and for negative (solenoids = -+)
Xs = linspace(-L/2, L/2, 2*steps);
Ys = linspace(-L, L, 2*steps);
Zs = linspace(.04, .07, steps);

Es = linspace(min_eccentricity, max_eccentricity, eccentricity_steps); %
    eccentricities evaluated
Volumes = zeros(eccentricity_steps, 7); % will contain the volumes and
    edges of the rectangoloids
eqs = zeros(length(Es), 1);

Fxs = zeros(1,length(Xs));
Fys = zeros(1,length(Ys));
Fzs = zeros(1,length(Zs));

h = waitbar(0);

%% Start searching
x0 = zeros(12,1);
sys = maglevSystem(x0, params, approximationType, 0);

for ec = 1:length(Es)
    Zs = linspace(.04, .07, 2*steps); % these parameters need to be reset
         to find eq_z
    Fzs = zeros(1,length(Zs));
    % searching for the new equilibrium to reset Zs
    for k = 1:length(Zs)
           temp = sys.f([0,0,Zs(k),zeros(1,9)]',zeros(params.solenoids.N
               ,1));
           Fzs(k) = temp(9);
    end
    [~,idx]=min(abs(Fzs)); % new equilibrium found
    eqs(ec) = Zs(idx);
    Zs = linspace(eqs(ec)-L/2, eqs(ec)+L/2, 2*steps); % resetting Zs
        around the new equilibria

    % calculating points along X axis where Fx is closest to 0
    x0 = zeros(12,1); x0(3) = eqs(ec);
    sys = maglevSystem(x0, params, approximationType, Es(ec));

    for i = 1:length(Xs)/2
```

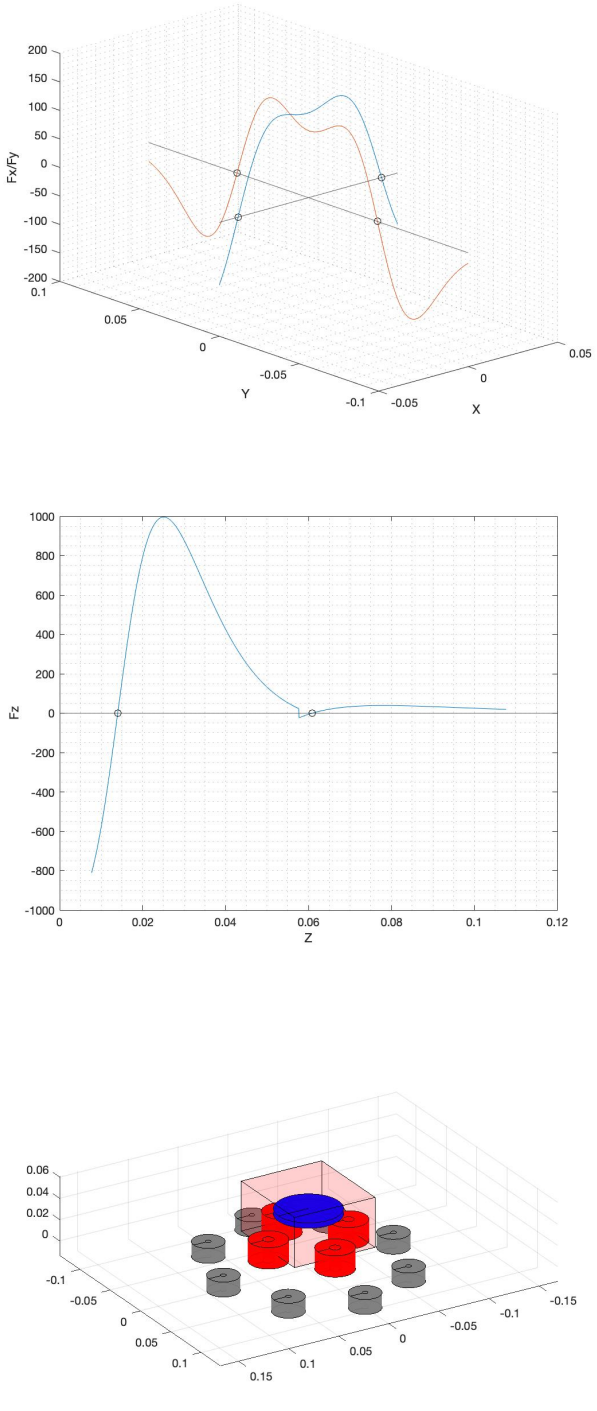
```
    x0(1) = Xs(i);
    temp = sys.f(x0,[.5 0 -.5 0]');
    Fxs(i) = temp(7);
end
for i = length(Xs)/2+1:length(Xs)
    x0(1) = Xs(i);
    temp = sys.f(x0,[.5 0 -.5 0]');
    Fxs(i) = temp(7);
end

% calculating points along Y axis where Fy is closest to 0
x0 = zeros(12,1); x0(3) = eqs(ec);
sys = maglevSystem(x0, params, approximationType, Es(ec));

for j = 1:length(Ys)/2
    x0(2) = Ys(j);
    temp = sys.f(x0,[0 .5 0 -.5]');
    Fys(j) = temp(8);
end
for j = length(Ys)/2+1:length(Ys)
    x0(2) = Ys(j);
    temp = sys.f(x0,[0 .5 0 -.5]');
    Fys(j) = temp(8);
end

% calculating points along Z axis where Fz is closest to 0
x0 = zeros(12,1); x0(3) = eqs(ec);
sys = maglevSystem(x0, params, approximationType, Es(ec));

for k = 1:length(Zs)/2
    x0(3) = Zs(k);
    temp = sys.f(x0,[.5 .5 .5 .5]');
    Fzs(k) = temp(9);
end
for k = length(Zs)/2+1:length(Zs)
    x0(3) = Zs(k);
    temp = sys.f(x0,[-.5 -.5 -.5 -.5]');
    Fzs(k) = temp(9);
end

% finding the indexes of the coordinates of the points where the
    components of F are smallest
idxs = zeros(1,6);
[~,idxs(1)] = min(abs(Fxs(1:end/2))); % smallest force for x < 0
[~,idxs(2)] = min(abs(Fxs(end/2+1:end))); idxs(2) = idxs(2) + length(
    Fxs)/2; % smallest force for x > 0
[~,idxs(3)] = min(abs(Fys(1:end/2))); % smallest force for y < 0
[~,idxs(4)] = min(abs(Fys(end/2+1:end))); idxs(4) = idxs(4) + length(
    Fys)/2; % smallest force for y > 0
[~,idxs(5)] = min(abs(Fzs(1:end*3/8))); % smallest force for z << eq
```

```
    [~,idxs(6)] = min(abs(Fzs(end/2+1:end))); idxs(6) = idxs(6) + length(
        Zs)/2; % smallest force for z > eq

    % dimensions of the rectangoloid
    [Volumes(ec, 1), Volumes(ec, 2)] = deal(Xs(idxs(1)),Xs(idxs(2))-Xs(
        idxs(1))); % vertex x, edge xl
    [Volumes(ec, 3), Volumes(ec, 4)] = deal(Ys(idxs(3)),Ys(idxs(4))-Ys(
        idxs(3))); % vertex y, edge yl
    [Volumes(ec, 5), Volumes(ec, 6)] = deal(Zs(idxs(5)),Zs(idxs(6))-Zs(
        idxs(5))); % vertex z, edge zl
    Volumes(ec, 7) = Volumes(ec, 2) * Volumes(ec, 4) * Volumes(ec, 6) * 1
        e6; % saving the volume in cm^3
    waitbar(ec/length(Es))
end

[~, idx] = max(Volumes(:,7));
fprintf("The eccentricity that grants the biggest controllable volume is:
     %f\n", Es(idx));
fprintf("Volume is: %f\n", Volumes(idx, 7));

close(h);

%% Plotting

% figures for a varying eccentricity
figure('Name', 'EQz / ecc');
plot(Es, eqs);
xlabel('Eccentricity'); ylabel('Height of the equilibrium')

figure('Name', 'Controllable Volume');
plot(Es, Volumes(:,7));
xlabel('Eccentricity'); ylabel('Volume')

%{
% figures for a single eccentricity
figure(1);
plot3(Xs,zeros(1,length(Xs)),Fxs); hold on;
plot3(zeros(1,length(Ys)),Ys,Fys);
xlabel('X'); ylabel('Y'); zlabel('Fx/Fy')
grid minor; xline(0); yline(0);
% rescaling X-Y axes to be equal, leaving Z axis as unvaried
h = get(gca,'DataAspectRatio');
if h(3)==1
     set(gca,'DataAspectRatio',[1 1 1/max(h(1:2))])
else
     set(gca,'DataAspectRatio',[1 1 h(3)])
end
plot3(Xs(idxs(1, 1:2)),0,0,'ko');
plot3(0,Ys(idxs(1, 3:4)),0,'ko'); hold off;
```

```
figure(2);
plot(Zs,Fzs); hold on;
xlabel('Z'); ylabel('Fz')
grid minor; yline(0);
Z = Zs(idxs(1,5:6));
plot(Z,0,'ko'); hold off;
%}
% plots the biggest (or only) rectangoloid
figure('Name', 'Controllability Region');
x0 = zeros(12,1); x0(3) = eq;
sys = maglevSystem(x0, params, approximationType, Es(idx));
hold on; draw(sys, 'fancy'); grid on; axis equal; view([45,30]);
plotcube([Volumes(idx, 2) Volumes(idx, 4) Volumes(idx, 6)], ...
        [Volumes(idx, 1) Volumes(idx, 3) Volumes(idx, 5)], .1, [1,0,0]);
             hold off;
```

As an example of the data obtainable with this script, the following images represent the aforementioned graphs of the forces plotted along the axes and the approximation of the controllable area for an eccentricity of 0.3 of both the permanent magnets and the solenoids.

Figure 3: Data acquired from a system of eccentricity 0.3



6.5 Condition Number Mapper

In order to get a more accurate (but more intricate to interpret) depiction of the controllable area, a second approach was tempted with the script CN_mapper.m, based on the evaluation of the condition number through a process presented by Berkelman and Dzadovsky (2010) [1].
Given the simple equation:

$$F_{nxm} a_{mx1} = b_{nx1} \tag{14}$$

the condition number of the matrix F, which is defined by the following equation:

$$\kappa(F) = \frac{|\lambda_{max}(F)|}{|\lambda_{min}(F)|} \tag{15}$$

can be consider as the rate at which a changes with respect to a variation of b. The evaluation of the condition number can thus be used to roughly estimate how much the state of the system will be altered as consequence of a change of the input of the system, with a low condition number meaning that the system is well controllable, and a high condition number meaning the opposite.
Although, due to the non-linearity of the system itself, a matrix F such that $F\dot{\eta} = u$ cannot be obtained from the ODEs of the system described in (10) and a linearization of the system was thus necessary. Since the input u affects only the last 6 components of $\dot{\eta}$, the linearized system can be simplified as follows:

$$\dot{\eta}_{7-12} = Fu \tag{16}$$

In order to obtain the relation, the desired F must then be inverted, but since it is a *6x4* matrix, the Moore–Penrose pseudoinverse is used instead. By linearizing and inverting the F matrix in several points of the space, it's possible to map the regions where its condition number is the lowest. A second version of the script was written called CN_mapper_fast; while the first version linearizes and maps the condition number in n^3 points of a cuboid, much like force_plotter.m, the second version only does so for the points on the XZ, YZ and XY planes containing the equilibrium, reducing the number of points to $3n^2$, thus greatly increasing performing times.

```
%{
    Linearizes the magnetic levitation system in every point of a cube
        and
    calculates the condition number
%}

clear; close all;
addpath('../maglevFunctions');
load('params.mat');
load('results.mat');

approximationType = input("approxType [0/1]> ");
```

```
eccentricity = input("Eccentricity [0:1]> ");
eccentricity = max (eccentricity, 0); eccentricity = min (eccentricity,
    1);

%% Searching parameters
steps = 150; % odd numbers will intersecate the equilibria in (0, 0),
    resulting in +infinity
L = .10;
bsqr = 1 / (1 - eccentricity^2);

if(approximationType == 0)
    eq = results.zeq.zeq_fst;
    params.magnets.I = results.neo_vs_neo.curr_fst;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_fst;
else
    eq = results.zeq.zeq_acc;
    params.magnets.I = results.neo_vs_neo.curr_acc;
    params.levitatingmagnet.I = results.neo_vs_lev.curr_acc;
end

%% Area to research
Ptx = linspace(-L/2, L/2, steps);
Pty = linspace(-L/2*bsqr, L/2*bsqr, steps); % assuming the ellipse's
    major axis is y
Zs = linspace(eq-L*9/20, eq+L*1/10, steps);
Cns = zeros(length(Ptx), length(Pty), length(Zs),1);

%% parameters for the linearization of system
x0 = zeros(12,1); x0(3) = eq;
sys = maglevSystem(x0, params, approximationType, eccentricity);

uLp = zeros(params.solenoids.N,1); % input (it doesn't affect the
    condition number, but it's needed to linearize)

delta = 1e-4;
dimX = 12;
dimU = params.solenoids.N; % 4

h = waitbar(0);
max = 3*steps;

B = zeros(dimX,dimU); % 12x4

%% Plane XY
%calculating k in this way gives an approximation of the nearest k to the
%equilibrium
k = round(length(Zs)*9/11); % chosen height of the plane to plot: around
    the equilibrium

fprintf("k: %f Zs(k): %f\n", k, Zs(k));
```

```
planeXY = zeros(length(Ptx), length(Pty)); % contains the condition
    numbers of the chosen XY plane
for i = 1:length(Ptx) %x
    for j = 1:length(Pty) %y
        % linearizing
        x0(1) = Ptx(i); x0(2) = Pty(j); x0(3) = Zs(k); % linearizing
        for l = 1:dimU % linearizing around this point
            B(:,l) = (sys.f(x0,uLp+(l==1:dimU)'*delta) ...
                    -sys.f(x0,uLp-(l==1:dimU)'*delta)) ...
                    /(2*delta);
        end
        temp = B(7:11, :); % coping only the non 0 rows
        Cns(i, j, k) = cond(pinv(temp)); % saving the condition number of
             the Moore-Penrose pseudinverse
        % saving the CN
        planeXY(i, j) = Cns(i, j, k);
        if planeXY(i, j) > 5e3 % for some reason max() gives an error
            if Cns(i, j, k) > 1e14 && ~(Ptx(i) == 0 && Pty(j) == 0)
                   fprintf("possible equilibrium at x: %f y: %f z: %f, or
                        i: %f j: %f k: %f\n", Ptx(i), Pty(j), Zs(k), i, j,
                         k);
            end
            planeXY(i, j) = 5e3; % saturating the results for better
                plotting
        end
    end
    waitbar(i/max)
end

%% Plane XZ
j = ceil(length(Pty)/2); % y = 0
fprintf("j: %f Pty(j): %f\n", j, Pty(j));
planeXZ = zeros(length(Ptx), length(Zs)); % contains the condition
    numbers of the chosen XZ plane
for i = 1:length(Ptx) %x
    for k = 1:length(Zs) %z
        % linearizing
        x0(1) = Ptx(i); x0(2) = Pty(j); x0(3) = Zs(k); % linearizing
        for l = 1:dimU % linearizing around this point
            B(:,l) = (sys.f(x0,uLp+(l==1:dimU)'*delta) ...
                    -sys.f(x0,uLp-(l==1:dimU)'*delta)) ...
                    /(2*delta);
        end
        temp = B(7:11, :); % coping only the non 0 rows
        Cns(i, j, k) = cond(pinv(temp)); % saving the condition number of
             the Moore-Penrose pseudinverse
        % saving the CN
        planeXZ(i, k) = Cns(i, j, k);
        if planeXZ(i, k) > 5e3 % for some reason max() gives an error
            if Cns(i, j, k) > 1e14 && Ptx(i) ~= 0
```

```
                fprintf("possible equilibrium at x: %f y: %f z: %f, or
                    i: %f j: %f k: %f\n", Ptx(i), Pty(j), Zs(k), i, j,
                     k);
            end
            planeXZ(i, k) = 5e3;
        end
    end
    waitbar((steps+i)/max)
end

%% Plane YZ
i = ceil(length(Ptx)/2); % x = 0
fprintf("i: %f Ptx(i): %f\n", i, Ptx(i));
planeYZ = zeros(length(Pty), length(Zs)); % contains the condition
    numbers of the chosen YZ plane
for j = 1:length(Pty) %y
    for k = 1:length(Zs) %z
        % linearizing
        x0(1) = Ptx(i); x0(2) = Pty(j); x0(3) = Zs(k); % linearizing
        for l = 1:dimU % linearizing around this point
            B(:,l) = (sys.f(x0,uLp+(l==1:dimU)'*delta) ...
                    -sys.f(x0,uLp-(l==1:dimU)'*delta)) ...
                    /(2*delta);
        end
        temp = B(7:11, :); % coping only the non 0 rows
        Cns(i, j, k) = cond(pinv(temp)); % saving the condition number of
             the Moore-Penrose pseudinverse
        % saving the CN
        planeYZ(j, k) = Cns(i, j, k);
        if planeYZ(j, k) > 5e3 % for some reason max() gives an error
            if Cns(i, j, k) > 1e14 && Pty(j) ~= 0
                fprintf("possible equilibrium at x: %f y: %f z: %f, or
                    i: %f j: %f k: %f\n", Ptx(i), Pty(j), Zs(k), i, j,
                     k);
            end
            planeYZ(j, k) = 5e3;
        end
    end
    waitbar((2*steps+i)/max)
end

close(h);

%% Plotting XY
figure('Name', 'XY Plane');
xlabel('X'); ylabel('Y'); hold on;
grid minor; axis equal;
[X, Y] = meshgrid(Ptx, Pty);
contour3(X, Y, planeXY, 'ShowText','on');
c.LineWidth = 3;
```

```
hold off;

figure('Name', 'XY Plane v2');
grid minor; axis equal;
[X, Y] = meshgrid(Ptx, Pty);
surf(Ptx, Pty, planeXY);
xlabel('X'); ylabel('Y'); zlabel('Cn')

%% Plotting XZ
figure('Name', 'XZ Plane');
xlabel('X'); ylabel('Z'); hold on;
grid minor; axis equal;
[X, Z] = meshgrid(Ptx, Zs);
contour3(X, Z, planeXZ.', 'ShowText','on');
c.LineWidth = 6;
hold off;

figure('Name', 'XZ Plane v2');
grid minor; axis equal;
[X, Z] = meshgrid(Ptx, Zs);
surf(Ptx, Zs, planeXZ.');
xlabel('X'); ylabel('Z'); zlabel('Cn')

%% Plotting YZ
figure('Name', 'YZ Plane');
xlabel('Y'); ylabel('Z'); hold on;
grid minor; axis equal;
[Y, Z] = meshgrid(Pty, Zs);
contour3(Y, Z, planeYZ.', 'ShowText','on');
c.LineWidth = 6;
hold off;

figure('Name', 'YZ Plane v2');
grid minor; axis equal;
[Y, Z] = meshgrid(Pty, Zs);
surf(Pty, Zs, planeYZ.');
xlabel('Y'); ylabel('Z'); zlabel('Cn')
```

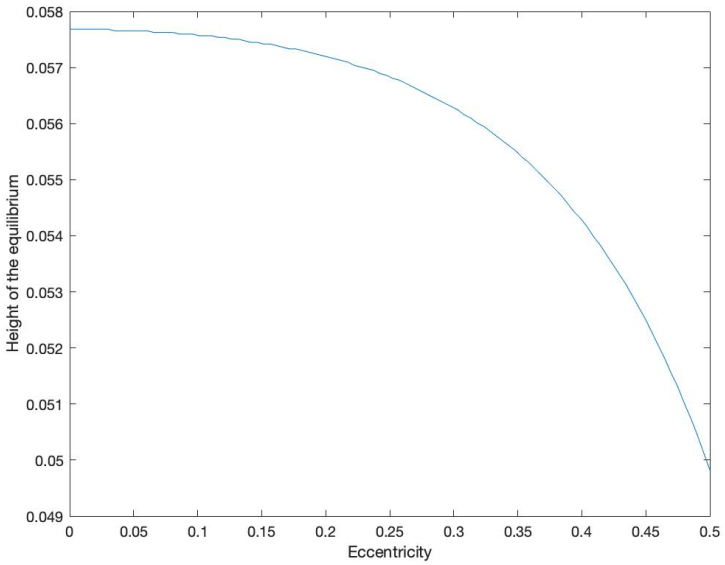
7 Discussion on the Results

The scripts described in the last chapter were used to analyze several parameters of the system and compare them between different elliptical arrangements of the permanent magnets and the solenoids. In particular it was observed how the height of the equilibrium and the volume of the controllable area would change in respect to an elliptical disposition of both the permanent magnets and the solenoids, an elliptical disposition of the permanent magnets and a circular disposition of the solenoids, and vice versa. The data of a circular system are also included for comparison.

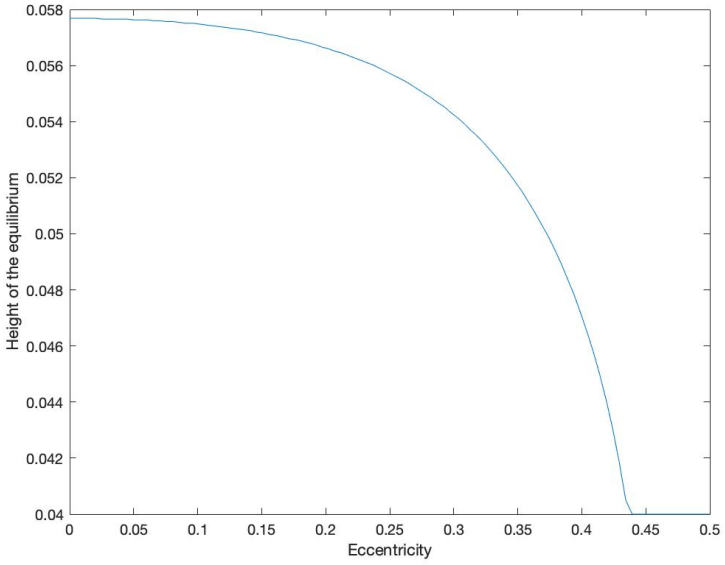
7.1 Height of the Equilibrium

The height of the equilibrium, as it was possible to predict, gets progressively lower with an increasing distance of the permanent magnets while is unaffected by the position of the solenoids. This is easily comprehensible since the equilibrium is calculated for an autonomous system, thus independent from the input. As a result, the height decreases as the eccentricity of the magnets increases, and for a circular design the equilibrium is even lower since the magnets are distancing along the x axis, too. As for what height is more favorable, it'll be discussed in the subsection relative to the condition number.

Figure 4: Elliptical vs Circular design

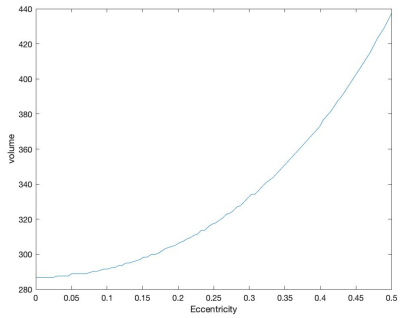


(a) elliptical disposition of the magnets

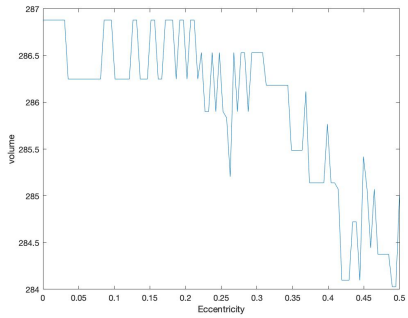


(b) circular design

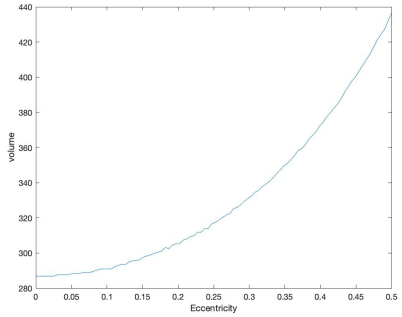
7.2 Controllable Area



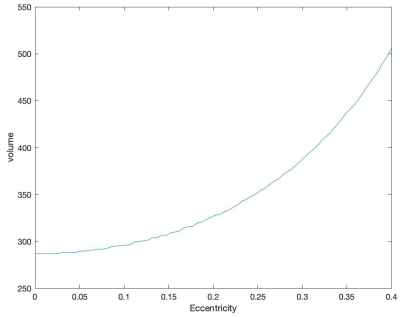
(a) elliptical disposition of the solenoids



(b) elliptical disposition of the magnets

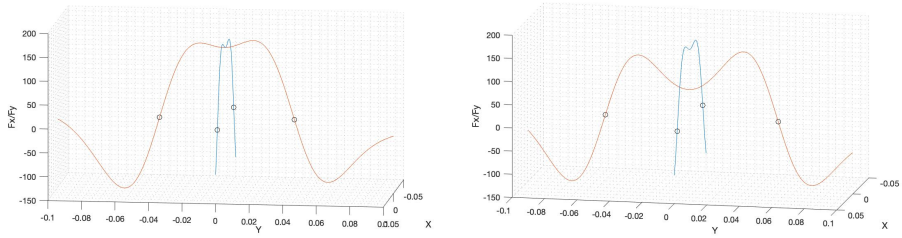


(c) elliptical disposition of the both



(d) circular design

These graphs refer to the volumes of the cuboid circumscribing the controllable area obtainable with the script eccentricity_controllability.m and several interesting things can be noted from them: first, it would seem that by increasing the eccentricity of both the solenoids and the magnets or the solenoids alone yields similar results, but a closer look at the numeric data reveals that in the first case the volume is consistently higher by a few cubic centimeters. this would seem to contradict figure (5b) which shows how an elliptic arrangement of the magnets actually reduces the ability to control the levitating disc.
In fact, the ellipticity of the magnets alone reduces the controllability, but it improves if paired with increasing the ellipticity of the solenoids, as seen by analyzing the measurements of the edges of the cuboid individually. It would seem that by increasing the eccentricity of the magnets, the height of the cuboid increases while the length and width decrease, but this last effect can be compensated by the ellipticity of the solenoids. In order to understand how the solenoids affect the controllability along the XY plane, it's possible to observe the graphs of the F_x and F_y forces described in section (6.4) and compare them for different eccentricities of the system.



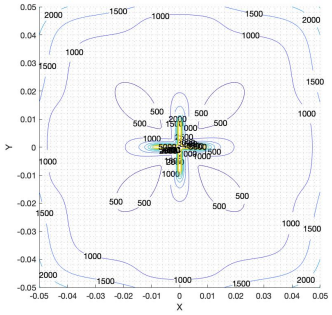
(a) circular design

(b) elliptical design

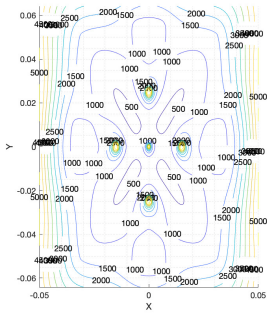
It is immediately noticeable how the increase in the volume of the controllable area is caused by the distancing of the points where F_y is close to 0. This also means that the strongest force applicable on the disk in the central zone of the platform is greatly reduced. These theoretical results thus suggest that a bigger controllable area comes at the cost of response time and reduced power to contrast errors.

7.3 Evaluation of the Condition Number

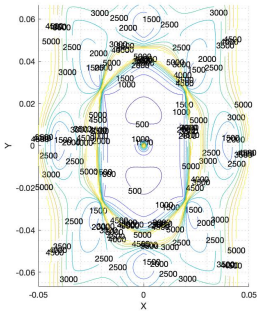
As mentioned before, eccentricity_controllability.m can only find the smallest cuboid containing the actual controllable area, while CN_mapper can calculate how "easy" it is to control the disc at any given position based on the value of the condition number of the linearized transformation matrix of the input of the system, so with it it is possible not only to obtain a more precise computation of the controllable area, but also to find where the controllability is "stronger", more stable, and disturbances of higher intensity may be negated.



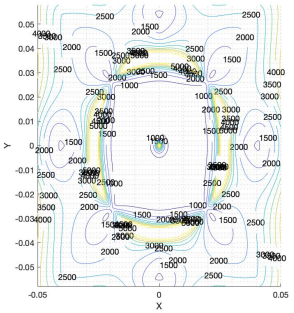
(a) Original circular design



(b) Eccentricity equal to 0.47



(c) Eccentricity equal to 0.5



(d) Greater circular design

The unstable equilibrium by definition has a condition number always equal to positive infinity, that is why each graph has a bright area in the middle, but the values around it can change greatly with the design of the platform. The bright area of the form of a cross around the equilibrium in the original design seems to confirm the hypothesis that the arrangement wasn't ideal, and by distancing the magnets of the platform it's possible to obtain a more or less consistent region where the condition number is relatively low. As these figures suggest, with a high enough ellipticity, the four low-conditioning regions of the original design merge into two along the ellipse's major axis, and these two regions wouldn't exist in a circular design with the same position of the equilibrium.

7.4 Conclusions

As it was expected, the results of this thesis don't show a particular design or a certain eccentricity as superior, but they highlighted the correlations between different characteristics of the system and how one can be increased at the expenses of another one. For example, they show how having the solenoids close to the center allows for a better control along the z axis but reduces the ability to control the disk along the XY plane, or how a bigger controllable area is possible only with a longer rising time.
In conclusion, the design should be chosen based on what trajectories the platform is intended to control, for example if it is only meant to keep the levitating magnet stationary at the equilibrium or if it needs to move it along a certain trajectory.

7.5 Future of the Project

There are of course many aspects of the project that can be improved by the next batch of students that will work on it, and hopefully this thesis will help them in finding what arrangement of the magnets better suits the goals they will have for the new platform.
Some possible adaptations were already presented in the new iteration of Engmark's description of the model regarding a different way of modelling the permanent magnets, moreover it could be tested if positioning the sensors in the spots where the flux of the magnetic field is null would be a more effective placement choice.

References

[1] Peter Berkelman and Michael Dzadovsky. "Novel Design, Characterization, and Control Method for Large Motion Range Magnetic Levitation". In: *IEEE Magnetics Letters* 1 (2010), pp. 0500104–0500104. DOI: 10.1109/LMAG.2009.2039341.

[2] Migdonio Alberto González and Dorindo Elám Cárdenas. "Analytical Expressions for the Magnetic Field Generated by a Circular Arc Filament Carrying a Direct Current". In: *IEEE Access* 9 (2020), pp. 7483–7495.

[3] Martin Brønstad et al. "Magnetic levitation system: Design, prototyping and testing of a digital PID-controller". B.S. thesis. NTNU, 2022.