Experimental investigations of a magnetic levitation system and the comparison with simulations

J.L. Beekman, s1091557
Supervisor: Dr. Will Robertson
Host organisation: University of Adelaide

November 4, 2016

Contents

Chapter 1

Acknowledgements

This report is the result of a research project about magnetic levitation systems carried out as an internship assagnment at the University of Adelaide under the supervision of dr. Will Robertson. The main personal purpose was to gain some valuable insight into real world problems and get some valuable work experience.

This all could not have been possible without the supervision of dr. Will Robertson, who inspired me to look beyond occuring problems and approach them from another way to solve them or to search for a workaround since there are many ways to Rome.
Also many thanks to Erich Tropeano from National Instruments and Matt Forbes, an undergrate student from the course mechanical engineering, who helped me out a lot when LabVIEW was broken (again).

Chapter 2

Introduction

Many modern systems, especially high precision mechanisms, are equipped with a vibration isolator. Vibration isolators remove undesired vibrations from a system or reduce their amplitude as much as possible which improves the performance of the system. These absorbers consist normally of a flexible support, for example a compression spring, while this research is focused on a different way to construct these vibration isolators. Magnets are used instead of springs and therefore a magnetic levitation system is achieved instead of a conventional vibration isolator. This was the subject of dr. Will Robertson's PhD Thesis. The goal of this research is to elaborate the theory and compare the experimental results with the simulations carried out by Daan Wilmink [6]. The outcome might then be the input for a future paper about this subject.

2.1 Quasi-zero stiffness

The dynamics of a system are changed when an vibration isolator is applied. In this, the stiffness plays an important role since it partly defines the natural frequency of the system. Unwanted vibrations can occur if the operation frequency is close or equal to the natural frequency of the system. Changing the natural frequency can therefore avoid unwanted vibrations, see figure 2.1. It can be seen that the vibrations at the operating frequency can be reduced to zero in this case.

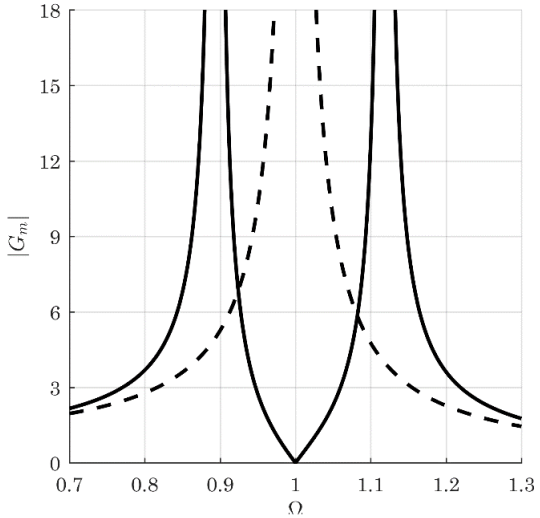


Figure 2.1: The dashed-line represents the frequency response function of a simple undamped mass-spring system with an natural frequency of $\omega_n = 1$ The solid-line represents the changed frequency response function after a vibration isolator is mounted with $\mu = 0.05$. Figure taken from [5].

The most ideal configuration for a vibration isolator is a configuration with no connection between the base and mass since there is no energy transfer possible between base and mass and therefore 'zero-stiffness' is achieved. However, this is not possible and therefore quasi-zero stiffness is defined as the inflection point of zero stiffness between positive and negative stiffness in a force versus displacement characteristic [4].

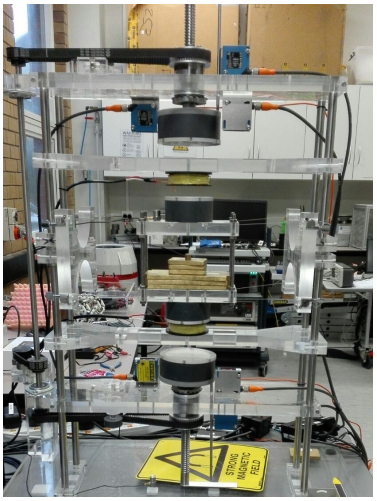
Nijsse [2] created different configurations which where able to construct a quasi-zero stiffness force-displacement relationship. To do so, a negative stiffness is added to a positive stiffness and therefore the natural frequency is adjusted. Consider again a simple mass-spring system but now with an added negative stiffness. The natural frequency simply becomes:

$$\omega_n = \sqrt{\frac{k_p + k_n}{m}} \tag{2.1}$$

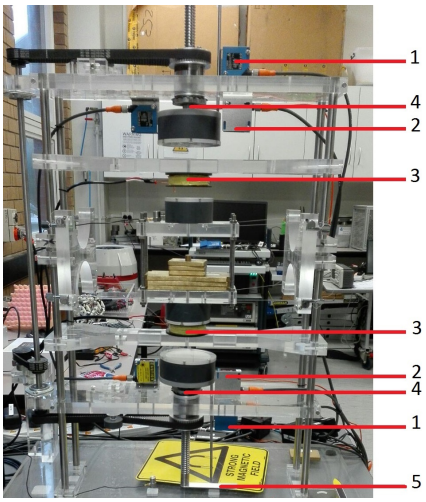
in which k_p is positive and k_n is negative. If $k_p = -k_n$ and linear springs are used, the natural frequency would decrease to zero. However, stiffness introduced by magnets are not linear and these non-linearities add complexity to the analysis and therefore quasi-zero stiffness is achieved instead of zero stiffness.

2.2 Magnetic levitation system set-up

Prior to the experiments a magnetic levitation system is designed to validate the simulations, see figure 2.2. Two pictures are shown: one of the set-up (figure 2.2a) and another one of the set-up with allocated numbers to it (figure 2.2b).



(a) Magnetic levitation system



(b) 1. Laser measuring distance to measurement plate; 2. Laser measuring distance to floating device; 3. Coil; 4. Load cell; 5. Measurement plate

Figure 2.2: Photograph of designed magnetic levitation system

The set-up consist of a main frame connected to a base. As one can see, two magnet pairs are shown in which the bottom one is in repulsion and the top one is in attraction. The floating device consist of a floating frame with mass in it connected to the main frame by guitar strings. Although one would prefer to create a non-physical connection so that the device is actually free floating in space (and therefore zero stiffness), this is not possible in the presence of gravity. Permanent magnetic levitation is unstable from itself according to Earnshaws' theorem. Strings are used to constrain the horizontal motion and therefore only vertical motion is achieved.

The top and bottom magnet are connected to a ball-screw mechanism to vary their vertical positions. A laser measures the relative distance to the measurement plate which is connected to each magnet construction. Four lasers measure the relative position in space of the floating mass. Since the lasers are not mounted in line, it is also possible to measure the rotation over both diagonals. Two coils are mounted to allow active feedback vibration control to further improve the isolation characteristics. The forces from the magnets exerted onto the floating device can be measured by two force sensors mounted between the magnets and the ball-screw mechanism.

The set-up is controlled by LabVIEW. A LabVIEW code are written by Matt Forbes, an undergrad student from the University of Adelaide, which should be capable of exciting the shakers, controlling the coils and displaying, logging and saving the data measured by charge amplified accelerometers and lasers.

2.3 Simulations

MATLAB scripts [3] were developed by Will Robertson to construct a force-displacement graph between a pair of magnets and thus determine the magnet stiffness given the dimension of the magnet and the gap (displacement) between the magnets.

Earlier experiments revealed the magnets as mounted in figure 2.2 did not behave as expected according to the theory. The theory is based on the assumption that the magnets have a constant and homogeneous magnetization however it is potentially difficult to produce magnets like these. New magnets were ordered to exclude this reason as the problem but these magnets should be validated before mounting them into the magnetic levitation system. This process will be explained extensively in chapter 4.

2.4 Structure of this report

Although LabVIEW is only used to control the system it plays an major role in this research. Therefore a bit more information is given in chapter 3. Chapter 4 is dedicated to magnets and the transmissibility is discussed in chapter 5. All this is used to draw conclusion and recommendations in chapter 6.

Chapter 3

Labview

In order to do proper experiments one should be capable of controlling the exciting frequency, measuring the distances between (moving) parts and determining the frequency response of the floating device and the base. This is done by LabVIEW in this project. By writing a LabVIEW program it is possible to accurately control the magnetic levitation system. However, this is quite complex to understand and therefore a brief introduction in LabVIEW is given in section 3.1 where all the components and separate layers of LabVIEW are discussed. Next the program(s) already written in LabVIEW will be discussed in section 3.2. Section 3.3 is dedicated to the modifications made to the programs.

3.1 Components of LabVIEW

LabVIEW is based on two components which can be divided in software and hardware which both consist of multiple layers. The software is the developing environment and within the software we can make a distinction between the 'Main User Interface (Main UI)', the 'Real-Time' and the 'FPGA'. The software is graphical programming approach based. Every layer has her own user interface on which all buttons, graphs and settings are displayed used in that particular component. The user interface is basically the visualization of the code, the rear side. The hardware, better known as 'NI-cRIO' (see figure 3.1) only consist of the layers Real-Time and FPGA and is programmed by the software. Multiple input and output devices can be inserted into the NI-cRIO for different purposes.

3.1.1 Main UI

The main UI is the user interface and the most important interface. All settings that have to be changed regularly should be on the user interface, for example exciting frequency and waveform. This allows the user to communicate with the magnetic levitation system and change the settings while doing experiments. The user interface itself only communicates with the Real-Time.

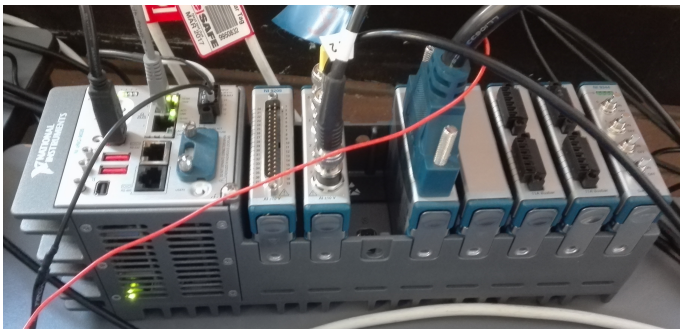


Figure 3.1: NI-cRIO 9035 with different modules

3.1.2 Real-Time

A Real-Time Operating System serves to manage real-time application data precisely and accurate while assuring consistent timing without losing important data. It is therefore able to communicate with the main user interface as well as the FPGA. Although the Real-Time runs at a high speed it is not as fast as the FPGA. The code for the Real-Time is designed in LabVIEW and then downloaded and executed onto the Real-Time device.

3.1.3 FPGA

FPGA is an abbreviation for Field Programmable Gate Array and are reprogrammable silicon chips [1]. This has an advantage over Application Specific Integrated Circuits (ASICs) since the latter ones are manufactured for one specific task only. An FPGA is ideal for developing a prototype since it can be reprogrammed. The FPGA runs at a high speed so it can achieve fast Input/Output (I/O) responses. To code the FPGA a program is designed in LabVIEW after which it can be compiled onto the FPGA. This process may take a long time depending to the amount of code to be compiled. The FPGA is always running after compiling while the Real-Time only runs when executed by user. The FPGA communicates with the I/O-modules assigned by the user (during coding) and with the Real-Time.

The communication between devices is summarized in figure 3.2. Modules used in the NI-cRIO are shown in table 3.1

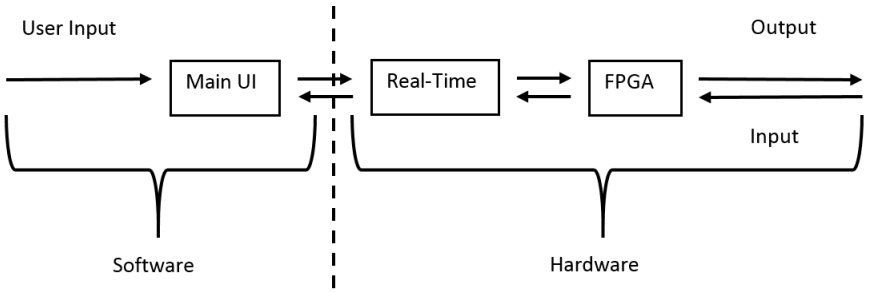


Figure 3.2: Communication between components of LabVIEW

Analog Input Module Old	NI 9205
Analog Input Module Final	NI 9239
Analog Output Module	NI 9264

Table 3.1: Modules used in the NI-cRIO

3.2 LabVIEW programs

An initial LabVIEW program to control the magnetic levitation system was written by Matt Forbes. The main UI is shown in figure 3.3. Next to the three green lights is the logging and control section. If the Real-Time is not responding or the program is not able to achieve the desired sample rate these lights will not switch on. If not, the actual sample frequency can be read from the box.

When the Real-Time is responding and the sample rate is achieved experiments can be carried out. Loggings can be started, stopped, saved an downloaded manually. This section also contains control buttons to excite the shaker, power the coils or switch on the motor.

The coil and the motor are additional components mounted on the device for research purposes in the near future but were not used in this project. First aim is to determine the transmissibility. Besides that, these buttons are not functioning at all so if one wants to incorporate this in the control the code should be adapted.

The section on the right, the grey box, contains all the adjustable settings of the magnetic levitation system. Information regarding waveforms, coil control, motor control, sampling rate and accelerometers are displayed and adjusted if necessary. Also information regarding the filenames is shown here.

The information from the Real-Time is shown in multiple graphs. Information shown here is not useful since the temporal resolution of the graphs is poor. However, this might come of use when validating the data roughly or debugging the system since accuracy is not important in those cases.

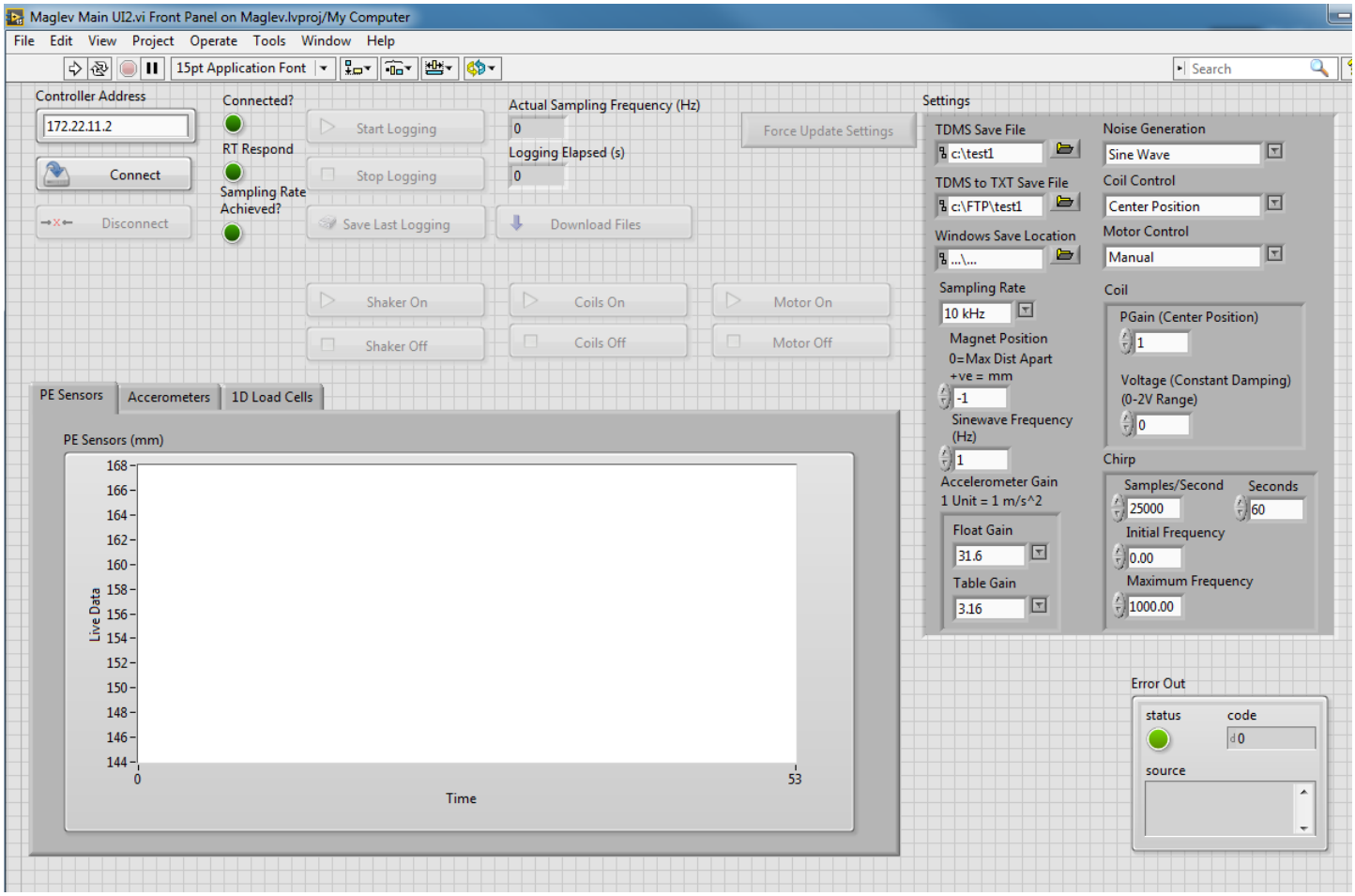


Figure 3.3: Main UI of LabVIEW program to control magnetic levitation system

3.3 Modifications

The LabVIEW program did not work when trying to excite the system at the first attempt. LabVIEW was not able to connect to the NI-cRIO and therefore no control was possible. Debugging was necessary to solve this problem but this task was quite hard because of all the different layers in LabVIEW. Not only the communication layers are existing, every block in the code was a new layer itself, a special function, within one of these three components. This function has his own user interface and block diagram which is separated from the user interface containing the function block. A structured overview may therefore be difficult to achieve which makes debugging even harder.

After solving the connection problem other errors arose. The cause of all these errors were reduced to the developing process with this final version as a result. Persistent errors forced the previous developer Matt Forbes to start over from a blank project. To avoid losing all work done previously the foundation of the code was copied into a blank project and then modified. Repeating this process several times have resulted in redundant functions.

These redundant functions were searching for connections that did not exist anymore and therefore end up in network errors. Other problems with the same cause and result are misaligned names. Although these problems are not to hard to fix generally, it is not easy to find these in a web of layers. Even the experts of NI had a hard time to locate this problem.

Solving this major problem was sufficient enough to use the LabVIEW program to control the system although minor problems were still present. A couple of tests were carried out to see of all the important components were working. This tests revealed that it is more convenient to control the run time of the shakers by the program instead of manually clicking the buttons. Therefore a new main UI is designed, see figure 3.4.

New tests indicated that the values measured by the NI-cRIO did not match up to the values measured by the oscilloscope. The values measured by the oscilloscope were a factor ten as large as the values measured by the NI-cRIO. The program was designed for a module that was capable of recording eight input channels. A hub was necessary to combine all inputs but this solution resulted into a untidy solution which is sensitive to errors.

After testing these connections the conclusions was drawn that the hub worked perfectly fine. The other possible problem might then be the functioning of the module itself. The only way to test this is to replace the module with a different one. Disadvantage of this is that switching modules results in a different FPGA due to the fact that the new module is only capable of recording four input signals instead of eight.

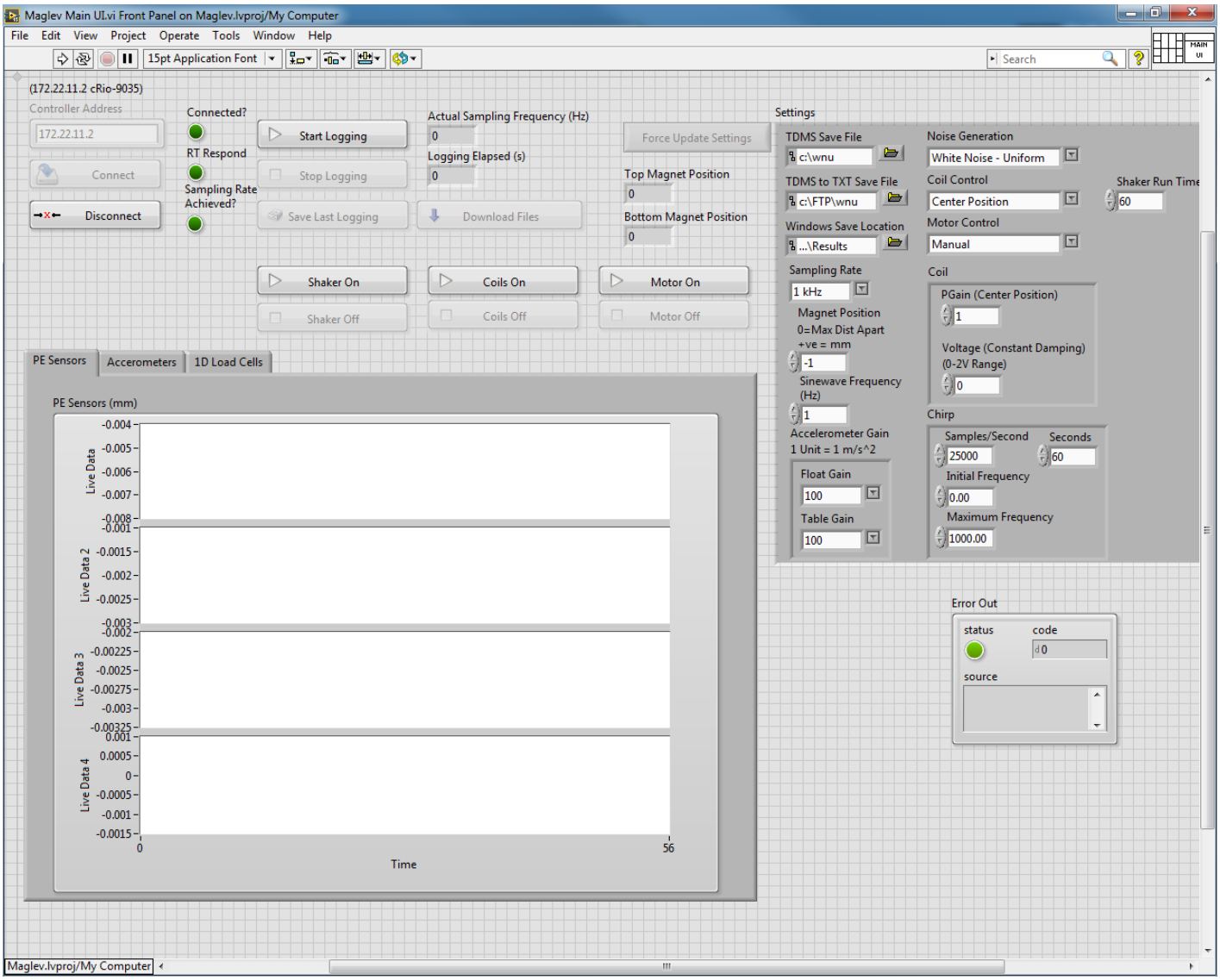


Figure 3.4: Modified Main UI in which the 'shaker run time' is added in the settings section. To have a more detailed look into the sensor values the sensors are shown separately instead of all together.

The advantage of using the new model instead of the old one is that the new module has an increased resolution and therefore an increased accuracy. While the old module was a 16-bit ADC, the new module is 24-bit ADC. This can be easily shown by calculation the Least Significant Bit (LSB):

$$\text{LSB} = \frac{V_{max} - V_{min}}{2^N - 1} = \frac{10 - -10}{2^{16} - 1} = 0.00030518043\ V \tag{3.1}$$

$$\text{LSB} = \frac{V_{max} - V_{min}}{2^N - 1} = \frac{10 - -10}{2^{24} - 1} = 0.00000119209\ V \tag{3.2}$$

The lower the LSB, the higher the resolution. Equation (3.1) is the equation for the LSB of the old module while equation (3.2) is the LSB of the new module. It is shown that there is a major difference in the LSB between the two modules and therefore in their resolution and accuracy. However, this solution did not fix the scaling problem between the NI-cRIO and the oscilloscope. In the end it turned out that there was a probe of factor ten set on the oscilloscope.

Although it was not necessary to recompile the FGPA in the end the choice is made to use the new configuration. Major advantage of this new configuration is the accuracy of the new module. Also, the overview is much better compared to the old one. However notice that there were still bugs available in the program but these bugs, see section 3.3.1, did not have any influence on the measuring results.

Final modifications are made to increase the accuracy of the tests even more. A shaker runtime was added to control the excitation time. The final main UI is shown in figure 3.5.

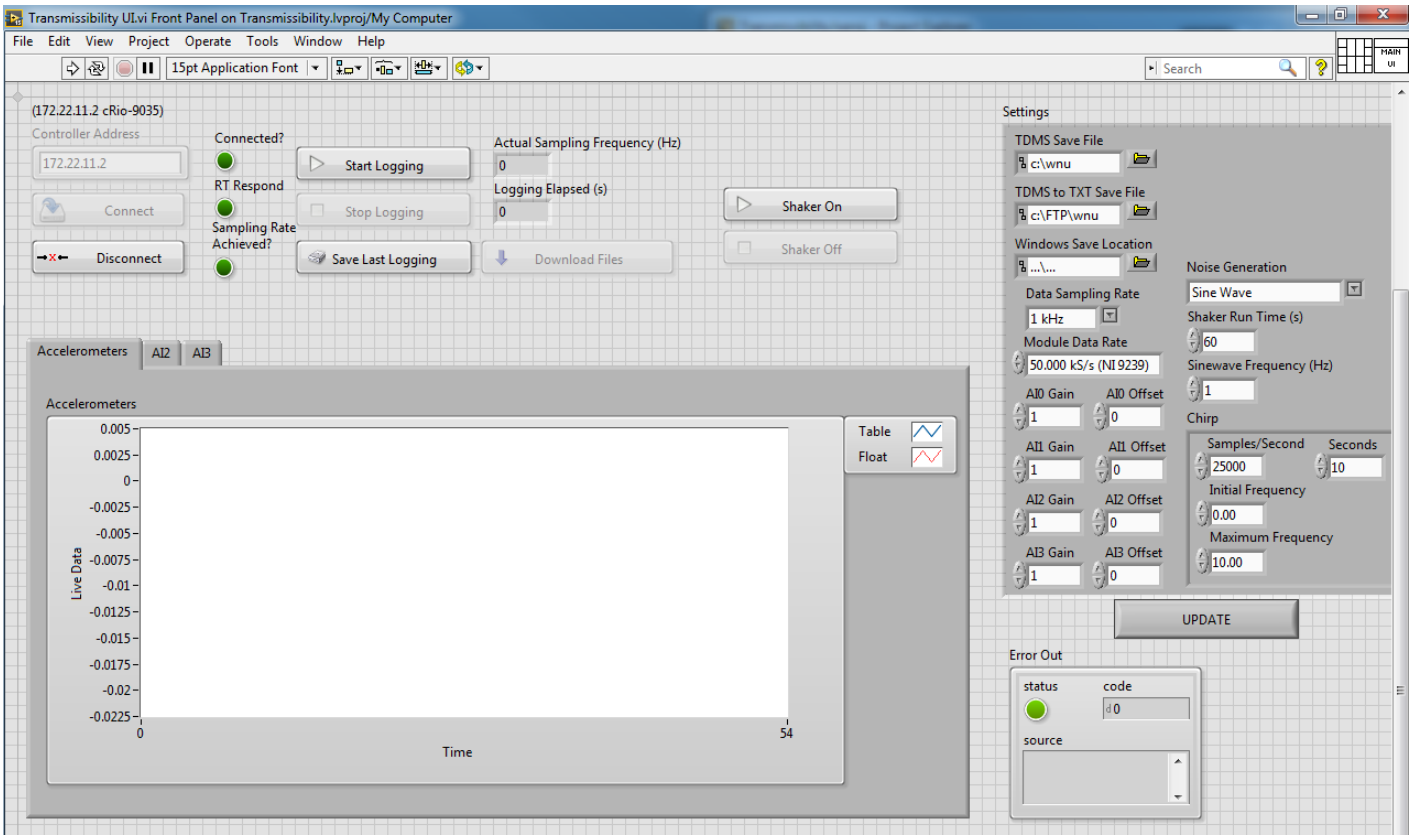


Figure 3.5: Final Main UI was created after swapping the modules. As a result of this, less inputs signals were recorded and therefore the graphs were adjusted to this module.

3.3.1 Bugs

Although the final main UI does work it is still far from perfect. It was capable of doing the experiments necessary to determine the transmissibility but a lot of bugs are still present. This might be caused due to a lot of redundant features. All bugs known will be listed here.

- This version of the program does work if one knows how to control the program. The most intuitive and easiest approach will cause a crash of the program due to a saving error. Therefore the correct sequence is given:

  1. Press start logging
  2. Press stop logging
  3. Look if LabVIEW shows a logged elapsed (s) value that is not equal to zero. If it is equal to zero the NI-cRIO and LabVIEW should be restarted. Then, start this sequence again.
  4. If the logged elapsed time is not equal to zero open the FTP of the NI-cRIO. Have a look into the c:/ folder of the NI-cRIO if LabVIEW created a filename equal to filename specified in the TDMS Save

File box. If not, restart the NI-cRIO and LabVIEW and start this sequence again. Continue if the file is in the c:/ folder.

5. Press start logging again
6. Press shaker on
7. The experiment is now carried out for the specified shaker run time. Press shaker off after the shakers stop exciting the system.
8. Press stop logging
9. Press save last logging
10. Press download files or download the data from the c:/FTP folder. If it is necessary to delete the data access the FTP folder from a computer other than the one on which LabVIEW is installed.
11. Restart the NI-cRIO and LabVIEW.
12. Do this sequence again if necessary

- It is necessary to check if the file is created before the actual experiment is carried out. LabVIEW might crash if the file is not created and if it does not, it is not recording data.
- It is necessary to restart the NI-cRIO and LabVIEW after the experiment is carried out. The shaker run time box does work but only for the first experiment. If a second experiment is carried out without a restart the shaker runtime is always less then the time specified.
- The logged elapsed time is not shown after an experiment is carried out
- Graphs shown are of low resolution. Only useful to check if values are approximately correct.
- This problem may or may not be related to LabVIEW but the amplifier lacks power when the system is excited with a sine frequency.

Chapter 4

Magnets

The forces exerted between a pair of magnets are three-dimensional which means there is a x-force, a y-force and a z-force. Determination of these force can therefore be quite complex although the x-force and the y-force are not of any interest in this research assuming there is no coupling between the forces while the system is in vertical vibrations. This is a valid assumption proven by Will Robertson [4]. A number of MATLAB scripts are created to make this calculation easier, see section 2.3. The MATLAB code can be found in appendix A.

4.1 Simulation

New magnets were ordered since previous magnets did not match up with the simulations. The new magnets were also cylindrical of shape with a radius of $r = 0.0375\ m$, $h = 0.030\ m$ and $grade = N42$. The fixed magnet has the same orientation as the floating magnet so this magnet pair is in attraction. A figure of the orientation is shown in figure 4.1.
By using the file *cyclmag.m* (code in appendix A.1) one is able to calculate the forces between a pair of magnets given the radius, height and strength of the magnets. The calculated force between the new magnets should be equal to the curve shown in figure 4.2.

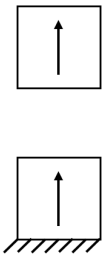


Figure 4.1: Magnet orientation to calculate force between magnet pair

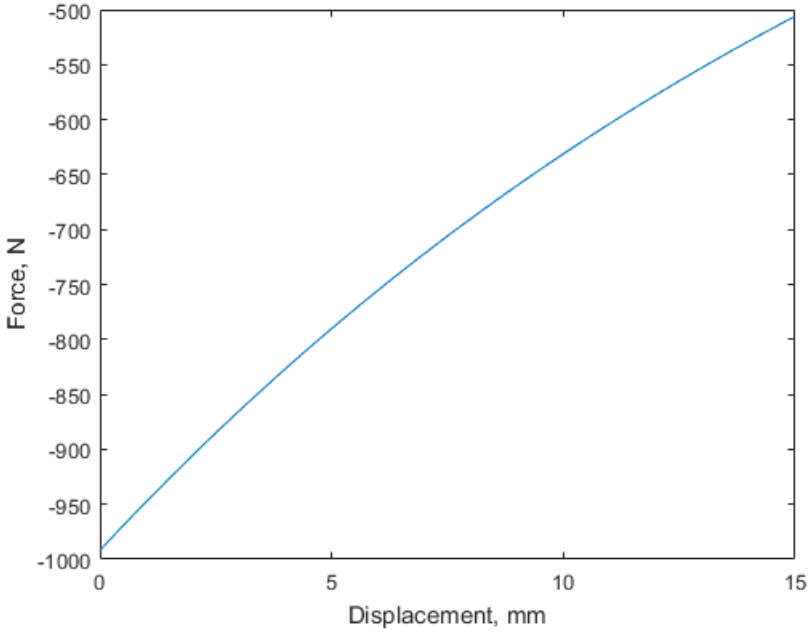


Figure 4.2: Results of forces between a pair of magnets in attraction.

As it can be seen with a gap of 5 mm between the magnets the force should be equal to 790 N. The larger the gap, the lower the force as the magnetic field is not strong enough to force the magnets to attract.

4.2 Validating

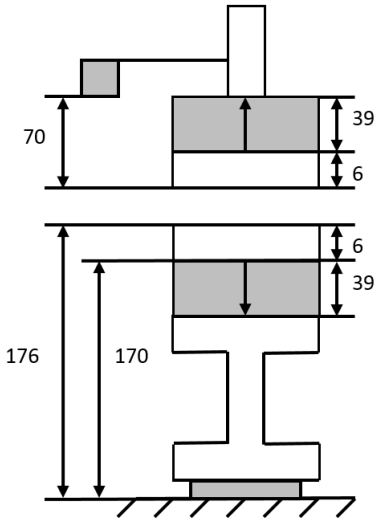
The new magnets should be validated before they are mounted into the magnetic levitation system. Ideally, the magnets do match the simulation but if not a curve fit of the actual situation should be made. This section describes the set-up of the measurement rig, the experiments and the outcome.

4.2.1 Set-up of magnet measurement rig

The magnet measurement rig is shown in figure 4.3a and the corresponding geometries of the magnet measurement rig are shown in figure 4.3b. A load cell is mounted on the bottom plate and attached to an I-profile which separate the bottom magnet from the load cell. This offset is necessary to attach the bottom magnet to the load cell since it is not possible to connect it directly to the load cell. Another advantage of this is that the strong magnetic field of the magnet might influence the electronics of the load cell and therefore the accuracy of the results.

The frame consists of four bars connected to the corners in which a slider is mounted. A laser is attached to the slider-bar and a magnet is mounted un-

(a) Measurement rig to determine force between a pair of magnets



(b) Configuration of magnet rig measurement set-up including geometries of rig

derneath the slider-bar. The magnets are mounted so that the magnets are in repulsion.

Once the slider bar is pushed down the gap between the magnets is decreased and this results in a stronger repelling force between the magnets. This force is measured by the load cell while the laser measures the difference in distance simultaneously. All data is logged and post processing is done in MATLAB afterwards.

4.2.2 Experiments

A number of settings should be checked before the experiments are carried out. The load cell is already subjected to a specific amount of weight because it has to support the mass of the I-profile and magnet. To calibrate the load cell the slider mechanism is removed and the force measured by the load cell is set to zero.

Since magnet forces in all directions are not independent it is important to check if the x- and y-force can be neglected with respect to the z-force. To do so ten experiments were carried out and compared. The result of one experiment is shown in figure 4.4.

It is important to note that the force in x-direction and y-direction is relatively small compared to the force in z-direction. Therefore the coupling between force can be neglected. Now that the assumption of an uncoupled z-force is proven to be valid other experiments were carried out, mainly to determine the force versus displacement graph.

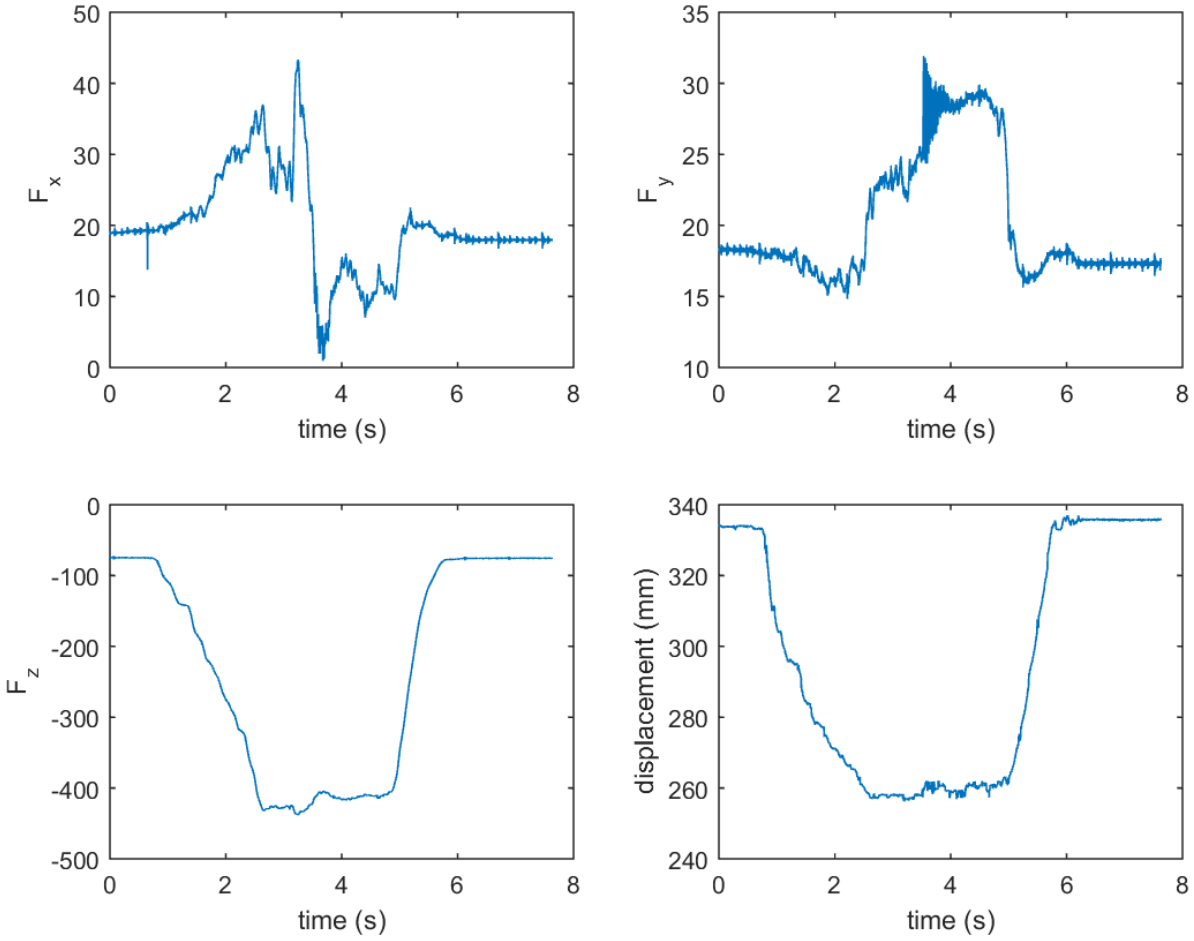


Figure 4.4: Result of one coupling experiment in which it can be seen that the x- and y-force can be neglected compared to the z-force.

Note that the graphs of the x-force and y-force are not smooth while the z-force is relatively smooth and also a lot of noise is observed in these measurements. The interesting part of these graphs are around 3.8 seconds. The amplitude of the x-force suddenly drops while the amplitude of the y-force suddenly increases surrounded by a lot of noise. Having a look at the z-force, nothing happens basically apart from a small decrease in amplitude.

Due to a statically overdetermined system the system the slider mechanism is subjected to a lot of friction. Even more, a lot of play is present in the magnet measurement rig. Because it was hard to push the slider straight down, small deviations occurs in the x-y-plane which causes features in the graphs.

Experiments to determine the force vs. displacement curves are carried out as follows:

1. Every magnet has a label from 1 to 4.
2. The bottom magnet, labeled as magnet 1 will always stay in place.
3. Carry out ten experiments between magnet 1 and magnet 2
4. Carry out ten experiments between magnet 1 and magnet 3
5. Carry out ten experiments between magnet 1 and magnet 4
6. Average results and check if all data is as expected and approximately equal to each other. If not, the bottom magnet should be replaced as well and all combinations should be investigated.

4.2.3 Disadvantages of magnet measurement rig

Although the outcome of the experiments is fairly good in terms of the z-force there are still a couple of options to improve the accuracy of the magnet measurement rig. A couple of options are summarized below:

- As discussed before the system is subjected to a lot of friction due to an statically overdetermined design. Because of the friction it is hard to push the slider mechanism straight down and small deviations occur in the x-y-plane. This problem can be solved by designing a new system which is statically determined. Also another 'slider-mechanism' should be used instead of wheels in a spline.
- The frictions varies from time to time. When mounting a new top magnet the slider mechanism is taken out of the main frame. Putting the slider mechanism back in place results in a different amount of friction because the configuration might be slightly different.
- The connection between the laser plate and the slider mechanism is not rigid which produces noise in the results.

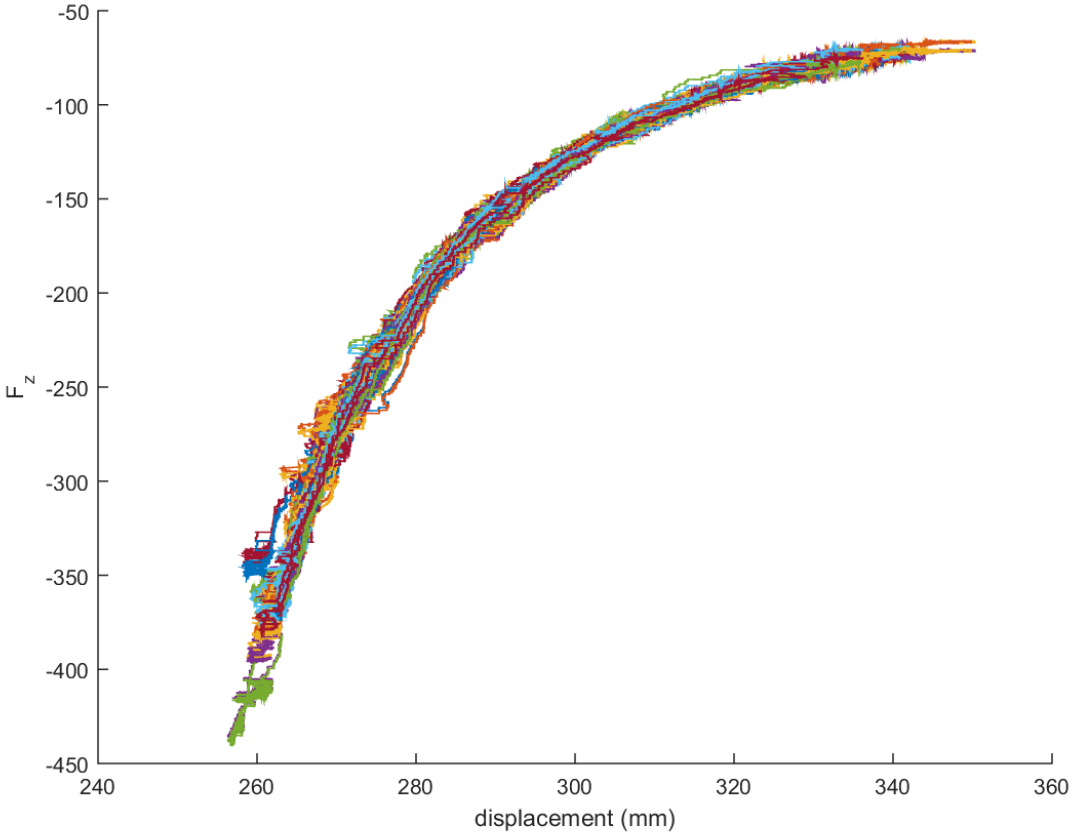
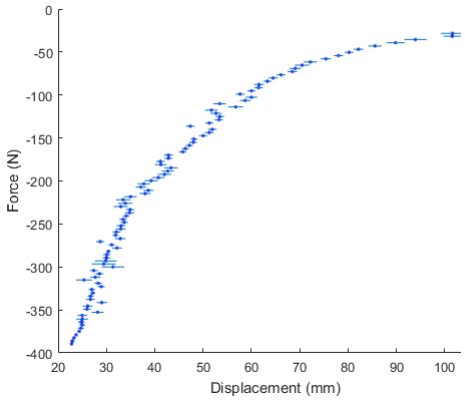


Figure 4.5: Sorted measurement data

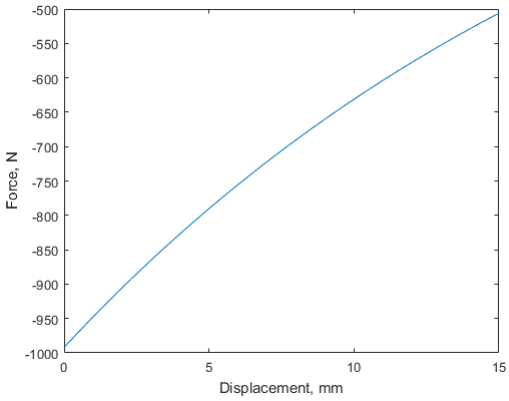
4.2.4 Results

After completing all experiments the data was loaded into MATLAB using *mag-postprocess.m* (code shown in appendix A.2) and analyzed. All data was combined into one major graph, see figure 4.5. As expected they should all yield almost the same result. This indicates that all magnets are more or less similar to each other. Although the magnets are similar to each other, no information is given if the magnets do meet the required specifications in accordance with the simulations.

First, all measurements are stored in two large arrays: one array of forces and one array of displacements. Then the lowest and highest value are located and the range is divided into a predetermined interval. Then find the nearest point to the interval point and correlate this point to its corresponding displacement. Take the average value of the forces and calculate the standard deviation in x-direction and y-direction. Repeat this step and the final result yields the average force-displacement curve, see figure 4.6a.



(a) Average force-displacement curve including standard deviation in x- and y-direction.



(b) Simulation of forces between a pair of magnets

Once the average force-displacement curve is determined it is easy to compare this curve to the simulations. For convenience the outcome of the simulation is repeated in figure 4.6b.

The format of both graphs is different but there is no need to adjust this. It can easily be seen that the actual forces are much lower than expected forces according to the simulations. When these magnets are mounted into the system a curve fit is necessary to control the outcome of future experiments rather than taking the simulation values.

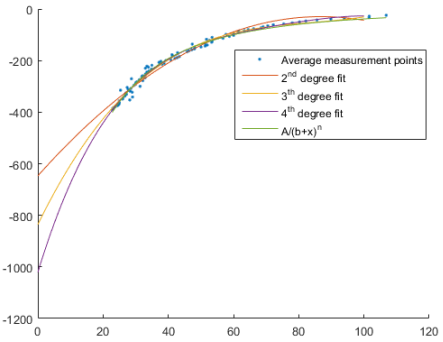
4.3 Curve fit

The data from figure 4.6a is taken as an input for the curve fit. Since the aim of this research is to compare the transmissibility of the actual device with the simulation, the same curve fit formula is chosen which is equal to:

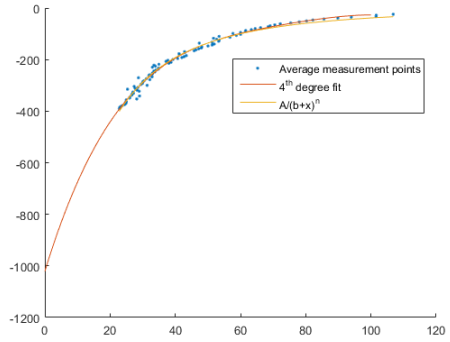
$$f(x) = \frac{A}{(x+B)^n} \tag{4.1}$$

Also a second degree polynomial curve fit, a third degree polynomial curve fit and a fourth degree polynomial curve fit are taken into account to make sure that equation (4.1) yields the best result, see figure 4.7a. The code used to carry out this curve fit is shown in appendix A.3.

It can be seen that the second and third degree polynomials do deviate a lot at higher displacements but only small deviations are allowed in this region since this is the operation range. Therefore only the fourth degree polynomial and equation (4.1) are analyzed into more details, see figure 4.7b.



(a) Average force-displacement curve including standard deviation in x- and y-direction.



(b) Simulation of forces between a pair of magnets

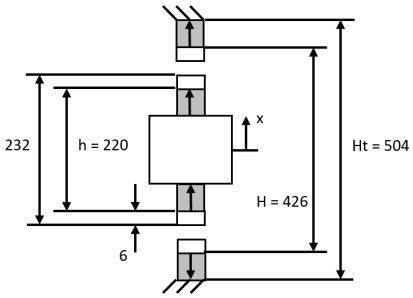


Figure 4.8: One possible configuration of the magnetic levitation system

Both curves follow the expected trajectory pretty close according to the average measurement points. However, in the operation region equation (4.1) does follow the trajectory better than the fourth degree polynomial. Therefore this is fit is chosen as the best fit. The final fit is equal to:

$$f(x) = \frac{-9.989 \cdot 10^6}{(x + 28.96)^{2.565}} \tag{4.2}$$

Now the fit is determined it is possible to create the force curve of the top magnet, the bottom magnet and the sum of these magnets and relate this to the mass of the floating device. In this way the equilibrium position is determined according to the measured data and this can then be validated by measuring the real design, see the code in appendix A.4.

One possible configuration to validate this is shown in figure 4.8. If the mass of the floating device is equal to 7.5630 kg the deviation of the center of the floating device from the static equilibrium position should be equal to -8.7 mm, see figure 4.9. When this is measured on the system the deviation is -9.6 mm.

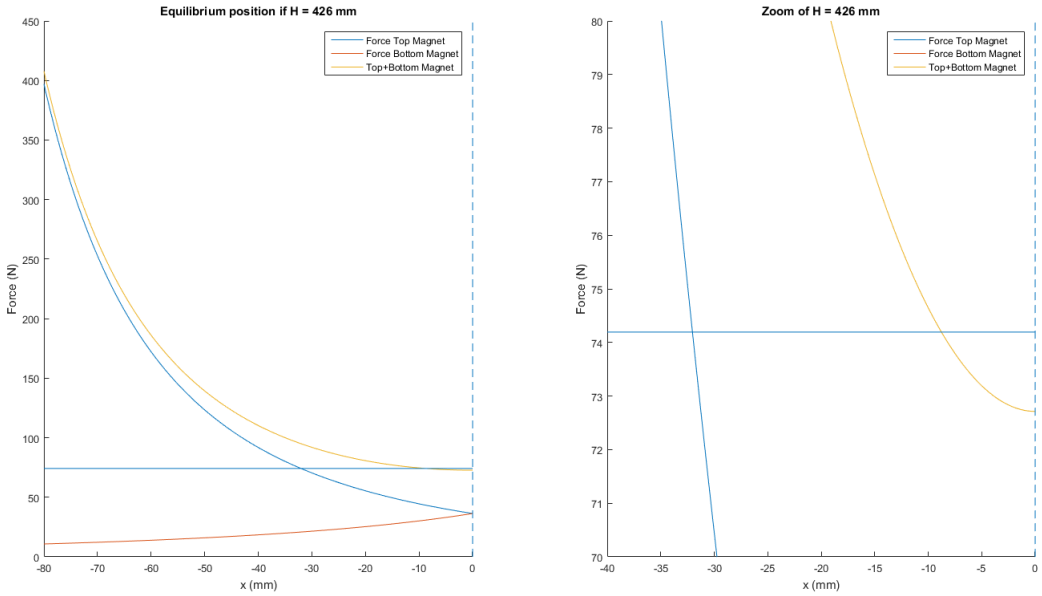


Figure 4.9: Equilibrium position according to measurements

The curve fit does not meet the real situation although the difference is small. One reason for this might be that the method to measure the distance on the magnetic levitation system was inaccurate. Because the module is changed from 8 inputs to 4 inputs it is not possible to use the lasers anymore. Therefore the distance was measured and calculated by hand but that is not accurate.

Later on the actual reason was discovered. While doing transmissibility experiments (see chapter 5) it became clear that the floating device did not have enough space to move. The coils were removed to achieve more space. It turned out that the screws used to mount the coil to the frame were not made of stainless steel and therefore have an influence on the magnetic field present.

New measurements should reveal in the future if this was actually the cause of the wrong equilibrium position.

Chapter 5

Transmissibility

In other to carry out the transmissibility experiments first the old magnets were swapped with the new ones. The force sensors were removed while swapping the magnets because the force sensors are not of any use at this stage. In fact, I've found out that one of the sensors is broken. Later on also the coils were removed since it turned out that the floating device did not have enough space to move up and down.

5.1 Experiments

Experiments of different times durations were carried out to ensure no data is lost while the time is preferably as short as possible. First, experiments of $t = 60\ s$, $t = 300\ s$ and $t = 600\ s$ are carried out to see if there is any difference between these experiments. If not, the most convenient option is to do further experiments with a time period of $t = 60\ s$.

The experiments are analyzed on four aspects. The time-domain signal is checked in order to see if enough energy is transferred into the system. If the signal is too low and transfer too little energy this will be visible in the time-domain graph. Also, clipping might occur which is not preferred and can be seen in this graph.

Then the power spectral density is investigated to see if only one clear peak is visible in the low frequency range as expected. The third aspect shows the transfer function while the last aspect shows the quality of the measurement.

At last multiple configuration of the magnetic levitation systems are checked while adjusting the distance between the top and bottom magnets and analyzed. Parameters used during these experiments are shown in table 5.1. The MATLAB-file used to determine the transmissibility can be found in appendix A.5

Sample rate	1000 Hz
FFT	2^{14}
Sample time	$t = 60\ s$, $t = 300\ s$, $t = 600\ s$
Average overlap	0.75

Table 5.1: Parameters used to determine the transmissibility

5.2 Discussion

When the time of the measurement increases better results are expected because more averages are used to calculate the frequency response. It turns out that the results did not meet these hypothesis, see figure 5.1 - figure 5.6. It can be seen that despite of the longer time measurement, no more significance is achieved compared to the time measurement of $t = 60\ s$. Instead of significance, more noise is introduced. Therefore only time measurements of $t = 60\ s$ are discussed since there is no sense to do longer experiments.

Figure 5.7 shows the time-domain signal. The time-domain displays that the signal is amplified as maximum as possible since the module has an measurement range of 10 V and still within limits because no clipping occurs. The top graph shows the time-domain signal of the base while the bottom graph shows the time-domain signal of the floating device.

The check of the time-signal did not reveal any unexpected observations which means it is possible to have a look at the power spectral density graph, see figure 5.8. Apart from the natural frequency the energy in the signal is fairly low. No well-founded conclusion can be drawn based on this because these low energy levels can contain a lot of noise.

The expected noise can indeed be seen in the transfer function, see figure 5.9. A smooth curve is expected but instead a chaotic curve is shown. This can all be explained by analyzing the coherence of the signal, figure 5.10. The value of the coherence is expected to be equal to one in this frequency range but instead the values are not even close to one and therefore the conclusion can be drawn that the quality of the measurement is very poor.

Although the outcome of the experiments is not satisfying it is possible to test the functioning of the system. Decreasing the gap between the bottom and top magnet of the magnetic levitation system should result in a lower natural frequency.

Figure 5.11 shows the result of an experiment with a decreased gap between the bottom and top magnets and the gap is even further in figure 5.12. It is indeed clearly visible that by decreasing the gap between the magnets the natural frequency decreases as well.

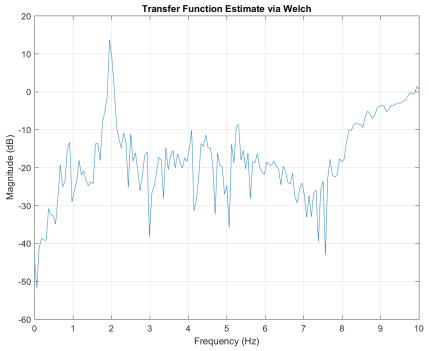


Figure 5.1: Transfer function, $t = 60\ s$

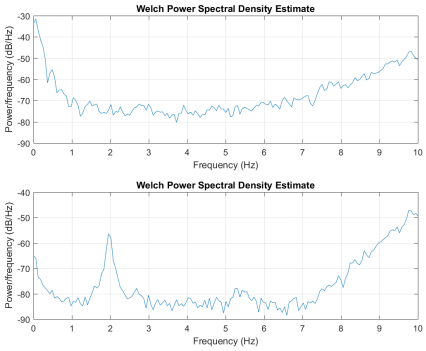


Figure 5.2: Pwelch, $t = 60\ s$

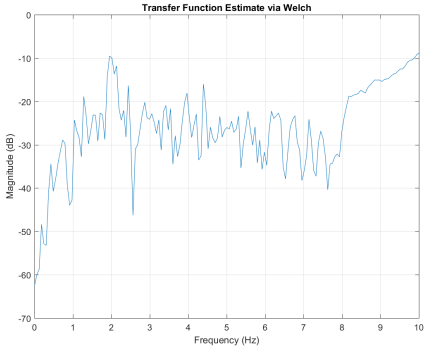


Figure 5.3: Transfer function, $t = 300\ s$

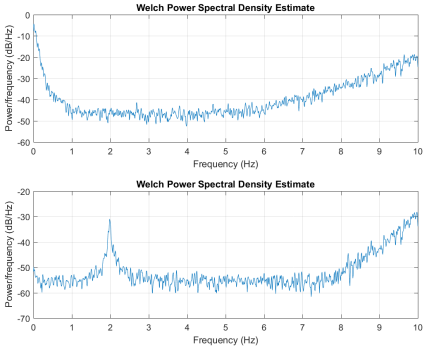


Figure 5.4: Pwelch, $t = 300\ s$

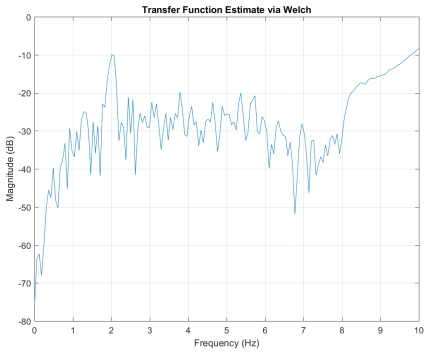


Figure 5.5: Transfer function, $t = 600\ s$

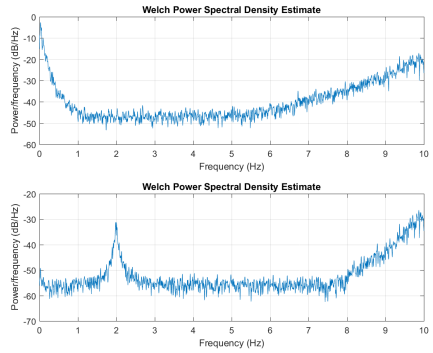


Figure 5.6: Pwelch, $t = 600\ s$

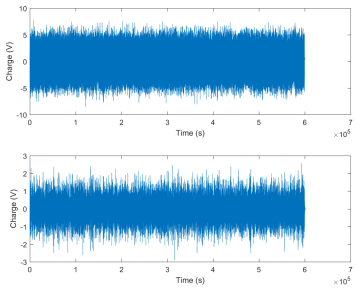


Figure 5.7: Time-domain, $t = 60\ s$

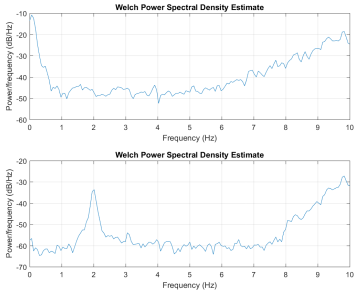


Figure 5.8: Pwelch, $t = 60\ s$

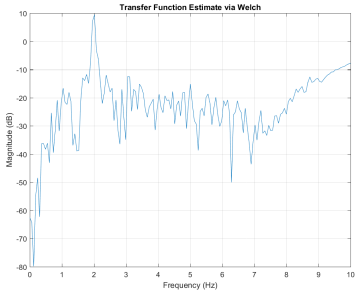


Figure 5.9: Transfer function, $t = 60\ s$

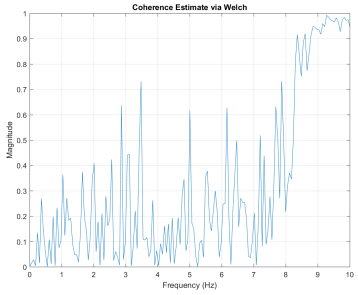


Figure 5.10: Cohere $t = 60\ s$

There is no point in comparing these results to the transmissibility simulations carried out by dr. Will Robertson due to a lot of noise in these experiments. During the experiments a lot of horizontal movement of the main frame was visible, so higher frequency dynamics of the system influenced the results a lot, see the power spectral density shown over a wider range of frequencies figure 5.13 and the transfer function, see figure 5.14. Also a lot of cable movement occurred especially in the cable of the accelerometer which may influence the results a lot. The magnetic levitation system is placed onto a table which also carries the equipment. When exciting the system the table is vibrating a lot as well as the equipment on the table.

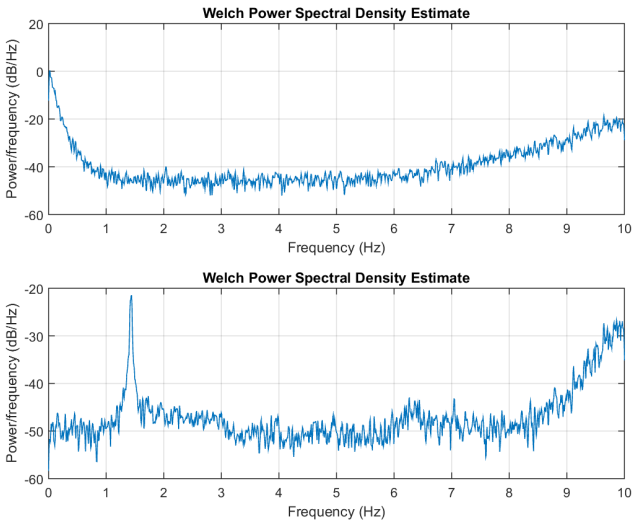


Figure 5.11: Decreased gap between bottom and top magnets compared to previous experiment shown in figure 5.8.

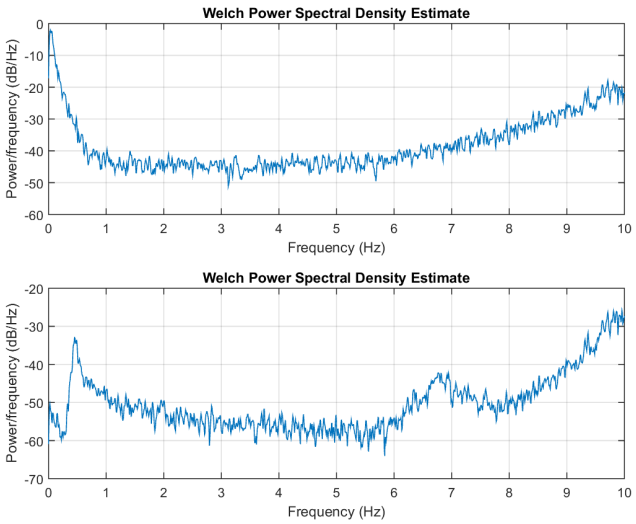


Figure 5.12: Distance between gap is again decreased compared to the experiments shown in figure 5.11.

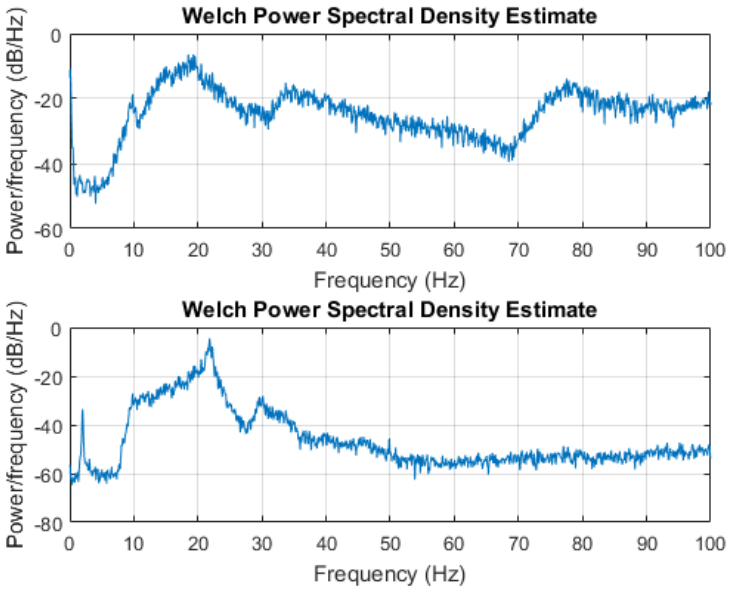


Figure 5.13: Power spectral density graph shown over a wider range of frequencies.

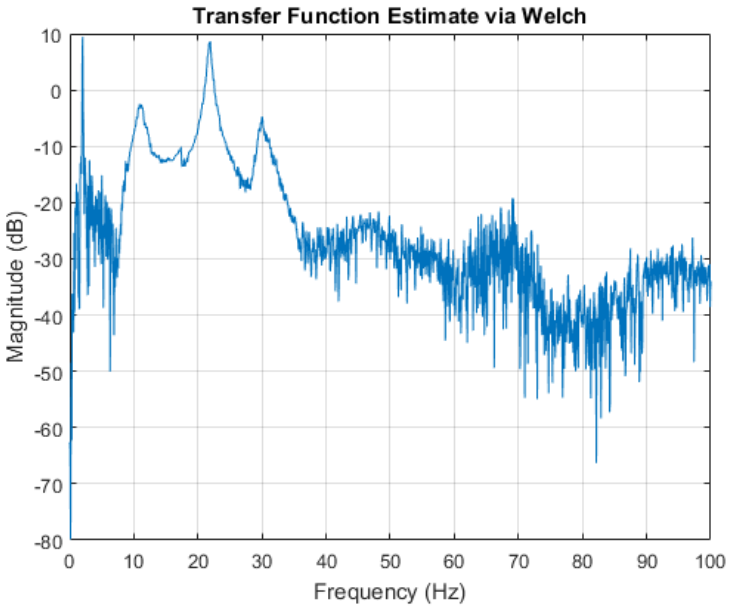


Figure 5.14: Transfer function of the system shown over a wider range of frequencies.

Chapter 6

Conclusion and recommendations

6.1 Conclusion

The aim of this research was to compare the measurements with the simulations done previously. Although the experiments did not yield the initial desired results, it showed that the system is potentially useful to reduce vibrations in high precision mechanisms once the problems with the excitation and measurements processes are fixed. A lot of possible options and configurations are explored, carried out and a lot of problems are exposed and can be improved in the future.

It is shown that in fact the magnetic levitation system can be used to decrease the natural frequency of a high precision mechanism and therefore remove vibrations from the system. However it is not possible to use this application straight away since more research is necessary to investigate this phenomenon into more detail.

6.2 Recommendations

Some recommendations can be done based on the experiments done to improve the system and investigate it in more detail.

1. It is shown that LabVIEW is capable of controlling the system. However a lot of redundant features causes a lot of errors and bugs in the program. In order to do proper experiments in the future the best way is to create a new program from scratch.
2. The magnet forces do not meet the simulated forces. More research is necessary to discover the source of this deviation between simulation and reality. Furthermore, the curve fit should be checked with the new configuration of the magnetic levitation system since the non stainless screws had their influence on the magnetic fields between the magnets.
3. The best way to avoid unwanted noise in the measurement data is to suspend the system and isolate the system from the equipment necessary

to carry out the experiment. Also more attention should be payed to higher frequency dynamics of the system and to the way the system is constructed. Possible sources of noise can be reduced by construction the system in a different way, for example the way the cables are connected to the system.

Appendix A

Matlab scripts

This appendix contains all important MATLAB files used in this research. It is divided in different sections in which one section contains one MATLAB-file complete with a brief description.

A.1 Cylmag.m

The script below is used to calculate the forces between cylindrical magnets. The radius, height of the magnet are necessary to calculate these forces as well as the magnetization. Also a direction vector should be provided in order to calculate the correct forces.

```
1  %% Matlab example for forces between cylindrical magnets/coils
2  %
3  % The script below uses the "magnetforces" function in the "matlab/"
4  % sibling directory. This function may be used to calculate the ...
       force
5  % between cylindrical and cuboid magnets with a common and ...
       consistent
6  % interface.
7
8
9  %% Define geometry and properties
10 %
11 % The magnet moves inside the coil and we want to calculate the ...
       force from
12 % the coil on the magnet.
13
14 % Magnet:
15 r2 = 0.0375; % metres
16 h2 = 0.030; % metres
17 J2 = 1; % Tesla
18
19 % % Coil:
20 % r1 = 0.02; % metres
21 % h1 = 0.02; % metres
22
23 % Displacement:
24 NN = 45;
25 displ_range = 0.045;
26 displ = linspace(0,displ_range,NN);
```

```
27
28 % Calculate forces:
29 fcyl = magnetforces(...
30   struct('grade','N42','dim',[r2 h2],'dir',[0 0 1]),...
31   struct('grade','N42','dim',[r2 h2],'dir',[0 0 1]),...
32   displ'*[0 0 1]...
33 );
34
35 figure(1)
36 plot(1000*displ,fcyl(3,:))
37
38 xlabel('Displacement, mm')
39 ylabel('Force, N')
```

A.2 Magnet Post Processing File

This scripts imports all data achieved by the magnet measurement experiments and structures the data for further research.

```
%% Initialize
close all
clc

mag12 = load('mag12.mat');
mag13 = load('mag13.mat');
mag14 = load('mag14.mat');
cg = load('cg.mat');                                    %Load data
lc = load('lc.mat');

% N = 5;                                              %Number of ...
    measurement files
% force = cell(1,N);                                  %Force cell
% displ = cell(1,N);                                  %Displacement cell
%
% for n = 1:N
% fid = 0;
%     while fid < 1
%         [fid,message] = fopen(['lc',num2str(n),'.dat'],'r');
%         if (fid == -1)
%             disp(message);
%         end
%     end
% C = textscan(fid,'%s %s %s %f');
% results = C{1,4};
%
% clear data % Not beautiful to clean data within a loop
% data(:,1) = results(4:8:length(results));   %Fx
% data(:,2) = results(3:8:length(results));   %Fy
% data(:,3) = results(1:8:length(results));   %Fz
% data(:,4) = results(2:8:length(results));   %Disp
% data(:,5) = (1:1:length(data))/10e3;        %Time
%
% %data = sortrows(data,3);
%
% force{n} = data(:,3);
% displ{n} = data(:,4)-170-64;
%
% end
%
% save('lc.mat')

%% Calculations separate measurements
%
% files = {'mag12','mag13','mag14','cg'};
%
% for file = 1:length(files)
%     if file == 1
%         force = mag12.force;
%         displ = mag12.displ;
%         N = mag12.N;
%         forceMeanMag12 = zeros(int-1,N);
%         stdxMag12 = zeros(int-1,N);
%         stdyMag12 = zeros(int-1,N);
%     elseif file == 2
```

```
%         force = mag13.force;
%         displ = mag13.displ;
%         N = mag13.N;
%         forceMeanMag13 = zeros(int-1,N);
%         stdxMag13 = zeros(int-1,N);
%         stdyMag13 = zeros(int-1,N);
%     elseif file == 3
%         force = mag14.force;
%         displ = mag14.displ;
%         N = mag14.N;
%         forceMeanMag14 = zeros(int-1,N);
%         stdxMag14 = zeros(int-1,N);
%         stdyMag14 = zeros(int-1,N);
%     elseif file == 4
%         force = cg.force;
%         dislp = cg.displ;
%         N = cg.N;
%         forceMeanCg = zeros(int-1,N);
%         stdxCg = zeros(int-1,N);
%         stdyCg = zeros(int-1,N);
%     end
%
%         % Declare variables
%         int = 100;                                          %Interval
%         point = zeros(int-1,N);                             ...
    %Interval points
%         lb = zeros(int-1,N);                                %Lower ...
    boundary
%         ub = zeros(int-1,N);                                %Upper ...
    boundary
%         minDifferenceValue = zeros(int-1,N);                ...
    %Difference value between point en existing point
%         indexAtMin = zeros(int-1,N);                        ...
    %Indicates minDiffrenceValue
%         Force = zeros(int-1,N);                             %Force ...
    array
%         Disp = zeros(int-1,N);                              ...
    %Displacement array
%
%         for n = 1:N
%             minval = min(force{1,n});
%             maxval = max(force{1,n});
%             spacing = (maxval-minval)/int;
%
%             start = minval;
%             for i = 1:(int-1);
%                 point(i,n) = start+spacing;
%                 lb(i,n) = point(i,n)-1;
%                 ub(i,n) = point(i,n)+1;
%                 start = point(i,n);
%             end
%
%             % Relate force to corresponding displacements and ...
    take the average of every
%             % point by using their boundaries. Then, take ...
    standard deviation in x-
%             % and y-direction.
%             Forcedata = force{n};
%             Displdata = displ{n};
%
%             for i = 1:(int-1)
%                 valueToMatch = point(i,n);
```

```
108 %
109 %                 % Find the closest force value.
110 %                 [minDifferenceValue(i,n), indexAtMin(i,n)] = ...
        min(abs(Forcedata - valueToMatch));
111 %
112 %                 %         fprintf('The closest value, %f, ...
        occurs at element %d.\n', ...
113 %                 %             Forcedata(indexAtMin(i,n)), ...
        indexAtMin(i,n));
114 %
115 %                 % Retrieve corresponding displacement
116 %                 Force(i,n) = Forcedata(indexAtMin(i,n));
117 %                 Disp(i,n) = Displdata(indexAtMin(i,n));
118 %             end
119 %
120 %             % Set interval by using boundary conditions (b.c.) ...
        Then, search for
121 %             % corresponding displacment according to lb & ub ...
        in orde to create
122 %             % horizontal standard deviation.
123 %             for i = 1:(int-1)
124 %                 findex=Forcedata<ub(i,n) & Forcedata>lb(i,n);
125 %                 forceBcArray = Forcedata(findex);
126 % %                 This code contains an error but can't find ...
        it! However,
127 % %                 the code does work when i = 10; If-statement ...
        below is
128 % %                 just for debugging.
129 % if file == 4
130 %     if i == int-2
131 %     str = 'stop';
132 %     end
133 % end
134 %
135 %                 dispBcArray = Displdata(findex);
136 %
137 %                 if file == 1
138 %                     forceMeanMag12(i,n) = mean(forceBcArray);
139 %                     stdxMag12(i,n) = ...
        std(dispBcArray,0,1,'omitnan');
140 %                     stdyMag12(i,n) = std(forceBcArray);
141 %                 elseif file == 2
142 %                     forceMeanMag13(i,n) = mean(forceBcArray);
143 %                     stdxMag13(i,n) = ...
        std(dispBcArray,0,1,'omitnan');
144 %                     stdyMag13(i,n) = std(forceBcArray);
145 %                 elseif file == 3
146 %                     forceMeanMag14(i,n) = mean(forceBcArray);
147 %                     stdxMag14(i,n) = ...
        std(dispBcArray,0,1,'omitnan');
148 %                     stdyMag14(i,n) = std(forceBcArray);
149 %                 elseif file == 4
150 %                     forceMeanCg(i,n) = mean(forceBcArray);
151 %                     stdxCg(i,n) = std(dispBcArray,0,1,'omitnan');
152 %                     stdyCg(i,n) = std(forceBcArray);
153 %                 end
154 %                 file
155 %                 i
156 %             end
157 %         end
158 %
159 %         % % Plot original displacement vs. force plot, just a ...
```

```
        check
160 %         % figure(1); hold on
161 %         % for n = 1:N
162 %         %     plot(Disp(:,n),Force(:,n))
163 %         % end
164 %         % hold off
165 %
166 %         % Plot displacement vs. avgerage force plot without ...
        changing displacements
167 % %         figure(2); hold on
168 % %         for n = 1:N
169 % %             plot(Disp(:,n),forceMean(:,n))
170 % %         end
171 % %         hold off
172 %
173 %         % Plot standard deviation
174 % %         figure(3); hold on
175 % %         for i = 1:(int-1);
176 % %             %     Relate avg force to displacement beforehand
177 % %             plot(Disp(i,n),Force(i,n),'b.')
178 % %             line([Disp(i,n)-stdx(i,n) ...
        Disp(i,n)+stdx(i,n)],[Force(i,n) Force(i,n)])
179 % %             line([Disp(i,n) Disp(i,n)],[Force(i,n)-stdy(i,n) ...
        Force(i,n)+stdy(i,n)])
180 % %         end
181 %
182 % end
183
184 %% All Raw Data combined an take the average
185
186 ForceArray12 = cell2mat(mag12.force');
187 ForceArray13 = cell2mat(mag13.force');
188 ForceArray14 = cell2mat(mag14.force');
189 ForceArrayCg = cell2mat(cg.force');
190 ForceArrayLc = cell2mat(lc.force');
191 ForceArray = [ForceArray12;ForceArray13;ForceArray14;ForceArrayCg];
192
193 DispArray12 = cell2mat(mag12.displ');
194 DispArray13 = cell2mat(mag13.displ');
195 DispArray14 = cell2mat(mag14.displ');
196 DispArrayCg = cell2mat(cg.displ');
197 DispArray = [DispArray12;DispArray13;DispArray14;DispArrayCg];
198
199 for i = 1:length(mag12.force)
200     N12(i,1) = length(mag12.force{1,i});
201 end
202
203 for i = 1:length(mag13.force)
204     N13(i,1) = length(mag13.force{1,i});
205 end
206
207 for i = 1:length(mag14.force)
208     N14(i,1) = length(mag14.force{1,i});
209 end
210
211 for i = 1:length(cg.force)
212     NCg(i,1) = length(cg.force{1,i});
213 end
214
215 % Declare variables
216 int = 100;                                      %Interval
217 point = zeros(int-1,1);                         %Interval points
```

```
218 lb = zeros(int-1,1);                                %Lower boundary
219 ub = zeros(int-1,1);                                %Upper boundary
220 minDifferenceValue = zeros(int-1,1);                %Difference ...
        value between point en existing point
221 indexAtMin = zeros(int-1,1);                        %Indicates ...
        minDiffrenceValue
222 ForceRaw = zeros(int-1,1);                          %Force array
223 DispRaw = zeros(int-1,1);                           %Displacement array
224
225 minval = min(ForceArray);
226 maxval = max(ForceArray);
227 spacing = (maxval-minval)/int;
228
229 start = minval;
230 for i = 1:(int-1);
231     point(i,1) = start+spacing;
232     lb(i,1) = point(i,1)-1;
233     ub(i,1) = point(i,1)+1;
234     start = point(i,1);
235 end
236
237 % Relate force to corresponding displacements and take the ...
        average of every
238 % point by using their boundaries. Then, take standard deviation ...
        in x-
239 % % and y-direction.
240 % Forcedata = force{n};
241 % Displdata = displ{n};
242
243 for i = 1:(int-1)
244     valueToMatch = point(i,1);
245
246     % Find the closest force value.
247     [minDifferenceValue(i,1), indexAtMin(i,1)] = ...
            min(abs(ForceArray - valueToMatch));
248
249     %         fprintf('The closest value, %f, occurs at element ...
            %d.\n', ...
250     %             Forcedata(indexAtMin(i,n)), indexAtMin(i,n));
251
252     % Retrieve corresponding displacement
253     ForceRaw(i,1) = ForceArray(indexAtMin(i,1));
254     DispRaw(i,1) = DispArray(indexAtMin(i,1));
255 end
256
257 % Set interval by using boundary conditions (b.c.) Then, search for
258 % corresponding displacment according to lb & ub in orde to create
259 % horizontal standard deviation.
260 for i = 1:(int-1)
261     findex=ForceArray<ub(i,1) & ForceArray>lb(i,1);
262     forceBcArray = ForceArray(findex);
263
264     %                 This code contains an error but can't find ...
            it! However,
265     %                 the code does work when i = 10;
266
267
268     dispBcArray = DispArray(findex);
269
270     forceMeanRaw(i,1) = mean(forceBcArray);
271     stdxRaw(i,1) = std(dispBcArray,0,1,'omitnan');
272     stdyRaw(i,1) = std(forceBcArray);
```

```
273
274 end
275
276 % Compensate for offset of load cell. Done by taking the average ...
        instead of
277 % a low-pass filter.
278 Offset = mean(abs(ForceArrayLc));
279 forceMeanRaw = forceMeanRaw+Offset;
280
281 %         Plot displaceme6nt vs. avgerage force plot without ...
        changing displacements
282 % figure(2); hold on
283 % plot(DispRaw,forceMeanRaw,'.')
284 % hold off
285
286 % % Plot standard deviation
287 figure(3); hold on
288 for i = 1:(int-1);
289     %     Relate avg force to displacement beforehand
290     plot(DispRaw(i,1),forceMeanRaw(i,1),'b.')
291     line([DispRaw(i,1)-stdxRaw(i,1) ...
            DispRaw(i,1)+stdxRaw(i,1)],[forceMeanRaw(i,1) ...
            forceMeanRaw(i,1)])
292     line([DispRaw(i,1) ...
            DispRaw(i,1)],[forceMeanRaw(i,1)-stdyRaw(i,1) ...
            forceMeanRaw(i,1)+stdyRaw(i,1)])
293 end
294 xlabel('Displacement (mm)')
295 ylabel('Force (N)')
296 save('disp.mat','DispRaw')
297 save('force.mat','forceMeanRaw')
```

A.3 Curve fit

This script creates a curve fit given the raw measurement data.

```
% CURVE FIT

%% Import data

Disp = load('disp.mat');
Force = load('force.mat');
disp = Disp.DispRaw;
force = Force.forceMeanRaw;
clear Disp Force;
m = 3.882;
mb = 3.75;
g = 9.81;

%% Polyfit

x = linspace(0,100,length(force))';

% p1 = polyfit(disp,force,2);
% p2 = polyfit(disp,force,3);
p3 = polyfit(disp,force,4);
%
f1 = polyval(p1,x);
f2 = polyval(p2,x);
f3 = polyval(p3,x);

%% Fit to A/(B+x)^n

f = fittype('a./(b+x).^n');
coeffnames(f);
yfit = fit(disp,force,f,'startpoint',[-1,-1,3],'lower',...
    [-10000000,-10,-10],'upper',[100,100,10]);

%% Plot
figure(1)
hold on
plot(disp,force,'.')
plot(x,f1)
plot(x,f2)
plot(x,f3)
plot(disp,yfit(disp))
legend('Average measurement points','2^{nd} degree fit','3^{th} ...
    degree fit',...
    '4^{th} degree fit','A/(b+x)^n')
hold off

figure(2)
hold on
plot(disp,force,'.')
plot(x,f3)
plot(disp,yfit(disp))
legend('Average measurement points','4^{th} degree ...
    fit','A/(b+x)^n');
hold off
```

A.4 Equilibrium

This script is used to determine the equilibrium position given the coefficients of the curve fit.

```
%% From CurveFit

CurveFit

% From curve fit
a = -9.989e6;
b = 28.96;
n = 2.565;

% Geometrical parameters
hm = 39;                    % Height of magnet casing without ...
    lid (mm)
Ht = 504;                   % Height from top top-magnet to ...
    bottom bottom-magnet (mm)
H = Ht - 2*hm;              % Height from center bottom magnet ...
    to center top magnet (mm)
h = 220;                    % Height floating devices + magnet ...
    (from center-to-center) (mm)
% For loop purposes

% Other properties
mf = 3.882;             % kg
m1 = 0.977;
m2 = 0.976;
m3 = m2;
m4 = 0.250;
m5 = 0.251;
m6 = m5;
m7 = m5;
m8 = 0.982;
m9 = 0.981;
m10 = 0.977;
g = 9.81;

% Determine
x = linspace(-80,0,1000);
d = (H-h)/2;
x1 = d-x;
x2 = d+x;

%% Equilibriium using A/(B+x)^n

ytop = -1*(a./(b+x2).^n);
ybottom = -1*(a./(b+x1).^n);
yt = ytop+ybottom;

% interp1((mf+m1+m2+m3+m4+m5+m6)*g,yt)

% Plot
figure(1)
set (gcf, 'Units', 'normalized', 'Position', [0,0,1,1]);
clf
subplot(1,2,1)
hold on
plot(x,ytop)
```

```
52  plot(x,ybottom)
53  plot(x,(ybottom+ytop))
54  % line([x(1) x(end)],[(mf+m1)*g (mf+m1)*g])
55  % line([x(1) x(end)],[(mf+m1+m2)*g (mf+m1+m2)*g])
56  % line([x(1) x(end)],[(mf+m1+m2+m3)*g (mf+m1+m2+m3)*g])
57  % line([x(1) x(end)],[(mf+m1+m2+m3+m4)*g (mf+m1+m2+m3+m4)*g])
58  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5)*g (mf+m1+m2+m3+m4+m5)*g])
59  line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6)*g ...
        (mf+m1+m2+m3+m4+m5+m6)*g])
60  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7)*g])
61  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8)*g])
62  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g])
63  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g])
64  line([0 0],[0 450],'LineStyle','--')
65  legend('Force Top Magnet','Force Bottom Magnet','Top+Bottom Magnet')
66  xlabel('x (mm)')
67  ylabel('Force (N)')
68  title('Equilibrium position if H = 426 mm')
69  hold off
70  subplot(1,2,2)
71  hold on
72  plot(x,ytop)
73  plot(x,ybottom)
74  plot(x,(ybottom+ytop))
75  % line([x(1) x(end)],[(mf+m1)*g (mf+m1)*g])
76  % line([x(1) x(end)],[(mf+m1+m2)*g (mf+m1+m2)*g])
77  % line([x(1) x(end)],[(mf+m1+m2+m3)*g (mf+m1+m2+m3)*g])
78  % line([x(1) x(end)],[(mf+m1+m2+m3+m4)*g (mf+m1+m2+m3+m4)*g])
79  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5)*g (mf+m1+m2+m3+m4+m5)*g])
80  line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6)*g ...
        (mf+m1+m2+m3+m4+m5+m6)*g])
81  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7)*g])
82  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8)*g])
83  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g])
84  % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g])
85  line([0 0],[0 450],'LineStyle','--')
86  axis([-40 0 70 80])
87  legend('Force Top Magnet','Force Bottom Magnet','Top+Bottom Magnet')
88  xlabel('x (mm)')
89  ylabel('Force (N)')
90  title('Zoom of H = 426 mm')
91  hold off
92  %
93  % print(['ABH',num2str(H)],'-depsc')
94
95  % %% Equilibrium using polyfit
96  %
97  % f1b = polyval(-p1,x1);
98  % f2b = polyval(-p2,x1);
99  % f3b = polyval(-p3,x1);
100 % f1t = polyval(-p1,x2);
101 % f2t = polyval(-p2,x2);
102 % f3t = polyval(-p3,x2);
103 %
```

```
104 % figure(2)
105 % hold on
106 % plot(x,f1b)
107 % plot(x,f1t)
108 % plot(x,(f1b+f1t))
109 % line([x(1) x(end)],[(mf+m1)*g (mf+m1)*g])
110 % line([x(1) x(end)],[(mf+m1+m2)*g (mf+m1+m2)*g])
111 % line([x(1) x(end)],[(mf+m1+m2+m3)*g (mf+m1+m2+m3)*g])
112 % line([x(1) x(end)],[(mf+m1+m2+m3+m4)*g (mf+m1+m2+m3+m4)*g])
113 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5)*g (mf+m1+m2+m3+m4+m5)*g])
114 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6)*g ...
        (mf+m1+m2+m3+m4+m5+m6)*g])
115 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7)*g])
116 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8)*g])
117 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g])
118 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g])
119 % line([0 0],[0 450],'LineStyle','--')
120 % legend('Force Top Magnet','Force Bottom Magnet','Top+Bottom ...
      Magnet')
121 % xlabel('x (mm)')
122 % ylabel('Force (N)')
123 % title('2^{th}')
124 % hold off
125 %
126 % figure(3)
127 % hold on
128 % plot(x,f2b)
129 % plot(x,f2t)
130 % plot(x,(f2b+f2t))
131 % line([x(1) x(end)],[(mf+m1)*g (mf+m1)*g])
132 % line([x(1) x(end)],[(mf+m1+m2)*g (mf+m1+m2)*g])
133 % line([x(1) x(end)],[(mf+m1+m2+m3)*g (mf+m1+m2+m3)*g])
134 % line([x(1) x(end)],[(mf+m1+m2+m3+m4)*g (mf+m1+m2+m3+m4)*g])
135 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5)*g (mf+m1+m2+m3+m4+m5)*g])
136 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6)*g ...
        (mf+m1+m2+m3+m4+m5+m6)*g])
137 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7)*g])
138 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8)*g])
139 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g])
140 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g])
141 % line([0 0],[0 450],'LineStyle','--')
142 % legend('Force Top Magnet','Force Bottom Magnet','Top+Bottom ...
      Magnet')
143 % xlabel('x (mm)')
144 % ylabel('Force (N)')
145 % title('3^{th}')
146 % axis([-80 0 0 400])
147 % hold off
148 %
149 % figure(4)
150 % hold on
151 % plot(x,f3b)
152 % plot(x,f3t)
153 % plot(x,(f3b+f3t))
```

```
154 % line([x(1) x(end)],[(mf+m1)*g (mf+m1)*g])
155 % line([x(1) x(end)],[(mf+m1+m2)*g (mf+m1+m2)*g])
156 % line([x(1) x(end)],[(mf+m1+m2+m3)*g (mf+m1+m2+m3)*g])
157 % line([x(1) x(end)],[(mf+m1+m2+m3+m4)*g (mf+m1+m2+m3+m4)*g])
158 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5)*g (mf+m1+m2+m3+m4+m5)*g])
159 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6)*g ...
        (mf+m1+m2+m3+m4+m5+m6)*g])
160 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7)*g])
161 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8)*g])
162 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9)*g])
163 % line([x(1) x(end)],[(mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g ...
        (mf+m1+m2+m3+m4+m5+m6+m7+m8+m9+m10)*g])
164 % line([0 0],[0 450],'LineStyle','--')
165 % legend('Force Top Magnet','Force Bottom Magnet','Top+Bottom ...
      Magnet')
166 % xlabel('x (mm)')
167 % ylabel('Force (N)')
168 % title('4^{th}')
169 % hold off
```

A.5 Transmissibility

This script determines the transmissibility of the system for three different experiments.

```
%% Initialize

close all
clc

%% Load data

for i = 1:3
    if i == 1
        file = load(['wnu60.mat']);
        N = 10;
        name = 60;
    elseif i == 2
        clear file
        file = load(['wnu300.mat']);
        N = 2;
        name = 300;
    elseif i == 3
        clear file
        file = load(['wnu600.mat']);
        N = 2;
        name = 600;
    end

    %%
    n = 1;
    for j = 1:N
        time = file.data_inp4{1,n};
        acc_bts = file.data_inp1{1,n};
        acc_fl = file.data_inp2{1,n};

        figure(1); clf;

        Fs = 1000;
        subplot(2,1,1)
        pwelch(acc_bts,[],[],[],Fs);
        xlim([0 100])
        subplot(2,1,2)
        pwelch(acc_fl,[],[],[],Fs);
        xlim([0 100])

        print([num2str(name),num2str(j),'Pwelch'],'-dpng')

        %%
        figure(2)

        subplot(2,1,1)
        plot(acc_bts)
        xlabel('Time (s)')
        ylabel('Charge (V)')
        subplot(2,1,2)
        plot(acc_fl)
        xlabel('Time (s)')
        ylabel('Charge (V)')

```

```
56          print([num2str(name),num2str(j),'Timedomain'],'-dpng')
57
58          %%
59          figure(3); clf;
60
61          Npoints = length(acc_bts);
62          Nfft = 2^14;
63          Nave = Npoints/Nfft;
64
65          Fs = 1000;
66          tfestimate(acc_bts,acc_fl,hanning(Nfft),round(0.75*Nfft),Nfft,Fs);
67          xlim([0 100])
68
69          print([num2str(name),num2str(j),'TFestimate'],'-dpng')
70
71          %%
72          figure(4); clf;
73
74          Nfft = 2^14;
75
76          Fs = 1000;
77          mscohere(acc_bts,acc_fl,Nfft,[],Nfft,Fs);
78          xlim([0 100])
79
80          print([num2str(name),num2str(j),'Cohere'],'-dpng')
81  %         close all
82          n = n + 1;
83      end
84  end
```

Bibliography

[1] National Instruments. What is an fpga? `https://www.ni.com/fpga/`.

[2] G.J.P. Nijsse. *Linear motion systems: A modular approach for improved straightness performance*. PhD thesis, Delft University of Technology, 2001.

[3] W.S.P. Robertson. Calculating forces between magnets using MATLAB. `https://github.com/wspr/magcode`.

[4] W.S.P. Robertson. *Modelling and design of magnetic levitation systems for vibration isolation*. PhD thesis, University of Adelaide, 2013.

[5] J.P. Schilder. Dynamics 2. Faculty of Engineering Technology, University of Twente, 2016.

[6] D. Wilmink. Theoretical and experimental investigations into magnetic levitation systems. University of Adelaide, 2015.