Development of Automatic Measurement System for Magnetic Field Distribution and Magnetization Estimation in a Permanent Magnet Based on Truncated Singular Value Decomposition

DOSHISHA UNIVERSITY

Graduate School of Science and Engineering

Department of Electrical and Electronic Engineering

International Science and Technology Course

Master’s Program

September 2022

Jared Jesus Davis

1316223301

Development of Automatic Measurement System for Magnetic Field Distribution and Magnetization Estimation in a Permanent Magnet Based on Truncated Singular Value Decomposition

September 2022

Jared Jesus Davis

1316223301

Development of Automatic Measurement System for Magnetic Field Distribution and Magnetization Estimation in a Permanent Magnet Based on Truncated Singular Value Decomposition

1316223301 Jared Davis Electrical Machinery Laboratory

1. Introduction

Ever since the world has adopted electronics in daily life it has become ever important to make the use of electricity on a large scale as efficient as possible. When it comes to electricity, magnetism is another side of the same coin. Being able to measure to accurately record properties of not only electricity but magnets, and induced magnetic fields allows for further efficiency. Being used on a massive global scale even the smallest change in efficiency could save a large scale of power in comparison. Not only being important for efficiency, but to safety and lifespan of electronics can be measured and recorded. Imperfect permanent magnets being used in machinery will deteriorate the devices they are used for [3].

Methods used currently for measuring magnetic fields include using probes to measuring not only the magnitude of the magnetic field, but also directional magnitude. This is important, because in some cases magnitude alone can in some cases not be enough to understand the health of the device. This method is done by manually measuring instances of the magnetic field and if done manually can sometimes include errors. Being able to create a system of measuring the magnetic field not only for instances, but for long periods of time can help remove this error and make the process much easier. In cases of a permanent magnet being able to visualize and measure the magnetic field is important, but another property is important that is the magnetism inside of the magnet. To obtain this magnetism one method is to transform the magnetic field using Biot-Savart law. With this transformation the magnetism can be found and visualized.

1. Measurement System

A 3D printer’s controller is removed and replaced with an Arduino UNO, this allows direct communication and control of it. A Lakeshore Tesla-Meter is then attached in place of where the filament extruded existed on the 3D printer. Communication to a main PC, the control the movement of the 3D printer and magnetic field measurement of the Tesla-Meter, these processes are done through a python code [11].

To control the motors using the Arduino a PCB is designed to control the individual motors each with its own limits switch to create a measurement area on the 3D printer bed.  A magnet is placed on the bed and the tesla-meter measures that said area for the magnetic field at and surrounding the magnet.

1. Magnetic Field to Magnetization

To find the magnetization at the magnet Biot-Savart law is used for its ability to obtain the Magnetization, ***M*,** from magnetic field, ***B*,** and vice versa. This comparison can be done at a given point *p* surrounding the magnet and in this case point *p* will be the measurement points of the magnetic field and at each point the magnetization will be solved for and using all these individual points *p*’s will be used to collectively solve for ***M***.

(1)

For every point that *p*, a matrix is given for the total number of measurement points. In this equation matrix ***K*** needs to be inverted to solve for ***M***. Using TSVD, Truncated Singular Value Decomposition, the pseudo-inverse can not only be solved for [5], but done properly can remove excess noise coming from the measurement process.

1. Results

To compare results a simulated magnet is constructed and using that magnetism, the magnetic field surrounding that magnet can be solved by doing the reverse process of solving for the magnetism. The same matrix ***K*** solved previously can be used for this process if the simulated magnet and magnetic field area are said to be the same.

First comparing the simulated magnetic field, Figure 1, to the measured magnetic field, Figure 2, the general shape and magnitude are very similar.

A graph of a magnetic field

Description automatically generated

Figure 1 Simulated Magnetic Field Z Direction

A graph of a magnetic field

Description automatically generated

Figure 2 Measured Magnetic Field Z Direction

This is a good start knowing that the measurement collecting system has done its job correctly. The next step is using TSVD and compare the magnetic field returned from the truncated magnetism to the simulated magnetic field.

 (2)

Truncating the data refers truncating the ***USV*** part of matrix ***K*** from previous SVD, seen in equation 2. *σ*k, refers to the singular values of matrix ***S***, from matrix ***K***. This data can be reduced not only for ease of use, but also for its ability to remove noise [5]. This is done by creating a threshold for these singular values inside of matrix ***S*** and reducing its overall rank, *r*, size. Doing so will further clean up the measured data to become as noise free as possible.

The noise can be seen in Figure 2 when looking to the area near and around the outside of where the magnet would have been placed. The noise occurs everywhere but is easily spotted in places where ideally the magnetic field would be 0.

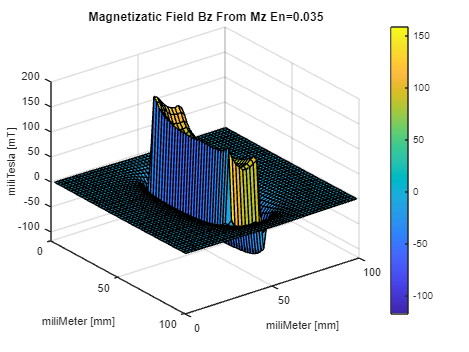


Figure 3 Magnetic Fieldεn=0.035 Z Direction

The singular values in matrix ***S***, are put through various values of thresholds, εn, and are tested to see how much data removed will remove the noise, but not remove too much data. Though with the rise of AI and Machine Learning, there are methods being tested to efficiently find good truncation values [6], it is still a very hard value to properly determine. Figure 3 captions one of the best values of εn, being 0.035, for having ideal data free from noise.

References

1. N. Nakai, Y. Takahashi, K. Fujiwara, and H. Ohashi, “Estimation of Magnetization Distribution in a Permanent Magnet using Genetic Algorithm,” *Proc. 17th Int. Symp. Electromagn. Fields Mechtron. Elect. Electron. Eng. (ISEF)*, Paper JP171, 2015.
2. L. Arbenz, O. Chadebec, C. Espanet, Y. Rtimi, and G. Cauffet, “Characterization of Permanent Magnet Magnetization,” *IEEE Trans. Magn.*, vol. 53, no. 11, Art. no. 8109504, 2017.
3. N. Nakamura, Y. Okamoto, K. Osanai, S. Doi, T. Aoki, and K. Okazaki, “Magnetization Estimation Method for Permanent Magnet Based on Mathematical Programming Combined with Sigmoid Function,” *IEEE Trans. Magn.*, vol. 58, no. 9, Art. no. 6000704, 2022.
4. S. Wakao, H. Igarashi, K. Fujiwara, A. Kameari, “Useful Formulas of Analytical Integration in Electromagnetic Field Computations (Part 1)” *Abstract Contents of Technical Report*, No.1043, 2006.
5. N. Nakamura, Y. Okamoto, K. Osanai, S. Doi, T. Aoki, and K. Okazaki, “Nondestructive Estimation of Magnetization Distribution in Permanent Magnet Using Quasi-Newton Method Based on 2-D Fourier Series” *IEEE Trans. Magn.*, vol. 56, no. 1, Art. no. 6000305, 2020.
6. M. Yeh “An Efficient, Tolerance-Based Algorithm for the Truncated SVD” *UC Berkeley Theses and Dissertations*, 2022.
7. H. Nishimi, Y. Ohnishi, Y. Takahashi, K. Fujiwara, “Visualizaition of Current Density Distribution in Photovoltaic Modules Using Truncated Singular Value Decomposition Based on Measurement of Surrounding Magnetic Flux Densities” *Doshisha University*, 2021.
8. D. Lausch, M. Patzold, M. Rudolph, C.-M. Lin, J. Fröbel, and K. Kaufmann, “Magnetic Field Imaging (MFI) of Solar Modules”, *Proc. of 35th EUPVSEC*, pp. 1060-1064, 2018.
9. H. Igarashi, T. Honma, A. Kost, “Inverse Inference of Magnetization Distribution in Cylindrical Permanent Magnets”, *IEEE Transactions On Magnets,* Vol. 36, No. 4, 2000.
10. PyCharm, JetBrains, 2024. [Online]. Available: https://www.jetbrains.com/pycharm/
11. LakeShore Python Library, Lake Shore Cryotronics, 2024. [Online]. Available: https://github.com/lakeshorecryotronics
12. PCB Design Software, Easy Eda, JetBrains, 2024. [Online]. Available: https://easyeda.com/
13. Arduino. 2024. Arduino IDE [Computer Software], Version 2.2.2. Available: https://www.arduino.cc/
14. M. McCauley, P. Wasp, “AccellStepper”, Version 1.64, Available: https://github.com/waspinator/AccelStepper, 2024

Development of Automatic Measurement System for Magnetic Field Distribution and Magnetization Estimation in a Permanent Magnet Based on Truncated Singular Value Decomposition

 Jared Jesus Davis

 1316223301

Development of Automatic Measurement System for Magnetic Field Distribution and Magnetization Estimation in a Permanent Magnet Based on Truncated Singular Value Decomposition

Admitted September 2022

Jared Jesus Davis

1316223301

Guidance by Yasuhito Takahashi

**Abstract**

Development of Automatic Measurement System for Magnetic Field Distribution and Magnetization Estimation in a Permanent Magnet Based on Truncated Singular Value Decomposition

By Jared Davis

Methods used currently for measuring magnetic fields include using probes to measuring not only the magnitude of the magnetic field, but also directional magnitude. This method is done by manually measuring instances of the magnetic field and if done manually can sometimes include errors. Being able to create a system of measuring the magnetic field not only for instances, but for long periods of time can help remove this human measurement error and make the measurement process easier. In cases of a permanent magnet being able to visualize and measure the magnetic field is important, but another property is important that is the magnetism inside of the magnet. To obtain this magnetism one method is to transform the magnetic field using Biot-Savart law. With some modifications to the equation, the magnetism can be found and visualized.

Using a 3D printer’s body and its motors and replacing its brain with an Arduino UNO, a Lakeshore tesla-meter is attached to where the extruder would have existed and designing the 3D printer to move around its printer area and syncing it with the tesla-meter a system for measuring a permanent magnets surrounding magnetic field is created.

An automatic measurement system allows for not only speedy, but safe collections of data. This data is visualized using Matlab and then using Biot-Savart law and TSVD the magnetic field, B is transformed to find the magnetism of the given permanent magnet. Simulations are constructed to find the ideal shape of the magnetic field, B and the magnetism, **M**, to use for comparison. Initial data collection could have remanent noise, so truncation is used, it is a very important process to help remove unwanted data.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| 1 Introduction ...………………………………………………………………….……. | 1 |
| 2 3D-Printer and Measurement System …………………………………………….… | 3 |
| 2.1 Ender 3D-Printer ..……………………….………………….…………….. | 3 |
| 2.2 Arduino UNO ...……………………….…………………….…………….. | 3 |
| 2.3 PC and PyCharm ……………..…………….……………….…………….. | 4 |
| 3 Arduino UNO …….………………………………………….……………………… | 5 |
| 3.1 Arduino Code …………………………………………….……………….. | 5 |
| 3.2 Arduino PCB ……………………………………………….………………. | 6 |
| 4 Measurement System ....…………………………………………………….………. | 9 |
| 4.1 Lakeshore Tesla-Meter ……………………………………….……….….. | 9 |
| 4.1.1 3D Printed Mount .…………………………………..………….. | 9 |
| 4.2 Python Code .……………………………………….……….…………….. | 9 |
| 5 Magnetization Estimation in a Permanent Magnet ……………………….………… | 12 |
| 5.1 Data Manipulation ………………………………………….…………….. | 12 |
| 5.2 Magnetic Field to Magnetization ….………………………….…….…….. | 12 |
| 5.2.1 Biot-Savart Law .………………………….……….…………….. | 12 |
| 5.2.2 TSVD for Matrix Solving…………………….…………………... | 13 |
| 5.3 Practical Application …………….……………………………….……….. | 14 |
| 6 Results ...…….. ……………………………………………….…………………….. | 16 |
| 6.1 Simulated Results ..………………………………………….…………….. | 16 |
| 6.2 Magnetic Field Results .……………………….………………………….. | 16 |
| 6.3 Magnetization Results ..…………………….…………………………….. | 17 |
| 7 Conclusions .……………………...………………………………………………… | 19 |
| References ……………………………………………………………………………… | 20 |
| Appendix A. LIST OF FIGURES AND TABLES …..…………………….................... | 21 |

1. **Introduction**

In the past couple hundred years the world has arguably seen the most significant evolution of technology ever in such a short period of time. The power of electricity has brought light to the night, connected every individual with a smart device, and has paved the way for even more progress that the world has yet to seen. This evolution is due the implantation of electricity. With electricity is equally important magnetism, two sides of the same coin that work like what seems to be magic to do feats that people a couple hundred years ago would believe to be impossible. The importance of understanding how electricity and magnetism not only work but how they affect their surroundings is of key importance to further growth and efficiency. This leads to the current research of surrounding magnetic fields.

Magnets and anything with electricity produce a magnetic field surrounding the object. Permanent magnets are commonly used in modern devices, seen with the rise of new technology like Electronic Vehicles EV, these permanent magnets are crucial to the functionality of the device. Imperfection in the magnetism of the permanent magnet will cause degradation of the device it is being used to function [3]. This field can be very delicate so understanding its depth, shape, and its change help are very important. Though from the magnetic field using other discoveries we can turn this magnetic field into other data to get insights on data other than the magnetic field. In the case where electricity is involved, for example, inside of a PV-cell the given magnetic field could be transformed to see the current density inside of the PV cell. With this information efficiency, faults, and degradation can be identified [2].

The process for this research contains two major sections. The data collection part and the post data portion taking the magnetic field data, ***B***, and transforming it to view the magnetization, ***M****,* using Truncated Singular Value Decomposition. The process for the data collection was inspired by a previous student at Doshisha University who was doing research on PV-Cells and PV-Modules using a similar process. The process for collecting the surrounding magnetic field involved using a sensor on the back of the panel and manually moving it from point to point [7]. The data was transformed to be able to see the current density distribution inside of the PV-cell to determine faults and productivity. This process though being accurate is not ideal from having a variable of possibility of errors from humans taking the data manually [8]. This inspired using assistance from a 3D printer.

The 3D printer would be very precise and consistent for taking data, which is very important when performing multiple data collections. The 3D printer also has a full range of movement whereas before all the data collections were only able to be taken in one plane, on the backside of the PV-cell. Although not be impossible it would be very difficult to take this data using a 3D printer do a variable of reasons, but most importantly physical limitations of having a PV-cell be place on the bed of a 3D printer. In this research case magnets are the target of the measurement so taking measurements on a bed of a 3D printer is very simple and has few physical limitations. This is very important for not only taking accurate measurements but also being able to take multiple measurements.

The 3D printer is operated remotely using an Arduino microcontroller to control the movements of the 3D printer and ensure the safety of the motors and experiment using limit switches to track the 3D printer beds location. The 3D printer has a Tesla-meter attached to the head of the 3D printer where the extruder would normally be for measurements. The attachment was 3D printed for the size of the Tesla-meter. The Arduino controller contains a designed PCB board that can control all the motors and switches.

The Tesla-meter is not controlled manually but by a PC designated for giving commands to both the 3d printer, via the Arduino, and the Tesla-meter. The Arduino attached to the 3D printer previously stated can move the 3D printer and control distance, but all commands come from the main PC which sends commands and collects data. This is all done via a program in Python. The program allows for simultaneous timing of the measurements and movements to ensure as much accuracy as possible.

After the data is collected the data is then processed and transformed into to find the magnetism, ***M***, of the Magnet. With this data it is then be presented to help give a clear understanding of the strength and shape of the magnetism of the given magnet.

**2 3D-printer and Measurement System**

**2.1 Ender 3D-Printer**

An inexpensive Ender 3D-printer was decided to be the most optimal option for this project with a large bed size and the ability to remove the motors connection and replace the brain of the printer with ease. The total bed of the 3D printer is 235 mm by 235 mm, but the motors can only move across on the center 220 mm by 220 mm’s of the bed. There have been multiple sizing changes of measurement area, but data measurements have occurred in a 150 mm by 150 mm area and 100 mm by 100 mm area. Using different areas depends on the target of the data collection. For example, in previous years when collecting data using 150 mm by 150 mm area was targeted for a PV cell that had a much larger area than the target magnet used in this research.

The 3D-printers’ movements were controlled by 3 stepper motors for each axis. The bed itself moves only in the *x* and *z* directions while the arm moves along the *x* axis. These motors are very accurate and have great responsiveness making them a great choice. Being made for 3D printing these motors also have engineered error prevention to get reliable results. Though being reliable the movement and repeatability of this movement time after time must be confirmed before continuing. In research like this exact precise repeatable movement is a must. Even a slight amount of being off will cause the data to be off making anything using this data will also be off compromising the research. This proved not to be an issue as all 3 directional motors were tested over a period of days to ensure that they have no issues when being used for this type of research. The testing involved repeatedly going back in forth directionally the same distance and measuring for inconsistencies.

**2.2 Arduino Uno**

For as much control of speed, distance, number of measurements, etc. it was decided that the original brain of the Ender 3D-printer was to be scrapped and replaced with an Arduino Uno. This Arduino Uno would allow for freedom of customization of movement and much more. For example, in the original data collection process the data was collected at each point in a 10 row by 10 column area. This proved to be too few collections and was needed to be tinkered around until a good amount of data could be collected. The same process happened when determining the proper area of measurement. Originally believing an overly large area was the best case, this proved to allow much more noise than anticipated and taking too much time for each measurement, so the area of measurement was slightly decreased. This was all done with very few adjustments thanks to the simplicity of changing a few lines of code in the Arduino’s IDE.

The use of the Arduino UNO also allowed communication between the measurement device of the magnetic field and the movement of the device. The Arduino acted to dish out directions for location and movement, keeping track of its internal location, while simply receiving directions like a remote controller. Simplifying the Arduino’s internal code allowed for any major changes to the number of collections and size of samples to be changed via the remote.

**2.3 PC and PyCharm**

For running the measurement collection process and sending signals to the Arduino a PC is used to allow communication. The PC does not need any major demands other than having more than one port to allow both the Arduino and the Lakeshore Tesla-meter to communicate. Python was chosen to be the language to run this code for reasons that will be explained in more detail later, but to run this Python script the PyCharm environment was used for its simplicity of use [10]. Lakeshore, the creator of the Tesla-meter being used in this research, also has a wide range of tools available for data collection in its python library [11]. These tools allow for the synchronization of measurements which is crucial to this research.

**3 Arduino Uno**

**3.1 Arduino Code**

The Arduino was powered by the USB connection of the PC during the process. One important thing to note is through this connection to the PC there are two programs being used. The original Arduino IDE[13] is used to upload code and has a serial connection to do testing and find errors. The main code for data collection is used by PyCharm, a community IDE for python coding. During the process of data collection, the Arduino cannot be connected to the Arduino IDE serial connection, because it is being used for data collection. This proved finding errors later in the programming process, when small data collection errors occurred, to be very difficult. The original plan was to have Arduino routinely send internal data through the data collection process between data collections. This allowed us to see at exactly what point in the data collection process things went wrong. This worked great until there were errors of the communication between the PC and Arduino occurred.

Before any and every data collection, the Arduino tells the 3D-printer to go through a calibration process. This process sets the *x, y, z* axis to 0 respectively so that once the measurement starts the code can tell where the data measurements occur and move them accordingly. The process is simple. The Arduino can accurately move each stepper motor easily. With this each motor during this process moves in the direction of its respective limit switch and once hit it has found its max and from there can be sent to the zero position and at that point the internal motor is told to set this location to 0. In this program the motors were controlled individually by the Arduino UNO output pins. Arduino IDE has some basic functions for motor control in its library [13], but for this type of movement with stepper motors to make ease of use a different library was implemented [14]. The library was specialized for controlling motor controls for instantaneous movements, acceleration, and long periods of time. Using that in tandem with limit switches means that if the switch is triggered this trigger can directly affect the motion of the stepper motors. This process ensures that through every data collection there is no worry of the stepper motors stepping out of range and damaging the 3D-printer and users, and to ensure that every collection is as accurate as possible. In the Arduino IDE with measurements of these stepper motors it was found for each 5 units of movement inside the code was equivalent to exactly 1 mm of movement for the stepper motors.

The original process for the movement of the 3D-printer involved all movements occurring from the Arduinos timing and simply matching that timing up with the measurements as done previously by a past student doing a similar experiment. This process had many reasons to be fixed like, not being completely accurate, not being able to change number of measurements without changing speed of the motor, and in cases where communication between the Arduino and PC occurred it is very difficult to tell at what point the error occurred.

This led to the original Arduino code being scrapped. The new code was designed to turn the Arduino into simply a receiver for directions. The reason for this was due to communication issues. It was discovered that in cases like this where communication is consistent and important it is best to use a looping code on the Arduino side. Previously bother codes worked independently and attempted to send information at important points, but by doing this even the slightest timing error meant that no information would be sent. In the new code, as previously stated, the Arduino now runs a continuously looping system. In each loop it simply looks to see if there are any new directions for movement. In those directions include its current position and where it needs to go. From there the Arduino simply stands on standby until the next movement.

The movement algorithm for the 3D-printer went through 3 different iterations throughout this research. Originally being told to go point to point in each measurement it was found that this meant that a change in the number of measurements this entire code would need to be rewritten every time. The original algorithm was created. The algorithm needed three things to be run the number of rows, columns, and number in the process. With this the code would divide its length by row to decide the size of each distance between measurements. For example, in the case of 150 mm in Arduino code this unit would be multiplied by 5 so that would be 750 and if the number of measurements per row is 30 then that would be in the Arduino code this means there are 25 units between measurements. Once all 30 collections were completed in one row the algorithm would reset it to that axis 0 point and then move over by 750 divided by the number of collections per column.

Using this algorithm the data collection worked great, but the data seemed to be shifted over from the center a couple of millimeters which did not make any sense in such an accurate data collection. Through some closer inspections to the data collection, it was found that when you start all the collections at 0,0 for the x and y axis and with the movements being length dived by number of measurements the code will never reach the true end of the area. This is because the code needs to not start at 0,0 but at the center between two of the lines calculated before. In the previous example 750 divided by 30 being 25 units, or 5 mm, including 0 there would need to be 31 measurements to reach 750 units. The issue was not that but where the data was being collected. These rows and columns create a grid on the area of the 3D printer bed, but it is not meant to be measured at these points, but the area created by those rows and columns being the center. With that in mind it was a simple update, by just changing the start position of each row and column and continuing as normal.

The next area works for 100 mm by 100 mm area. With 50 by 50 rows and columns there are a total of 2500 measurements being taken for each *x, y, z* direction of the magnetic field. The code calibrates and puts the starting position at 1 mm from the zero origin since the distance between measurements is 2 mm starting it at 1 mm will put it right in the middle for the ideal measurements.

Towards the end of writing this code it became a possibility that other students would eventually use the current 3D-printer method to measure their own data and continue their research. To make this process easier, the final movement code of the Arduino was updated to allow for complete freedom of the area of the measurements and the number of measurements. To do this the variables seen at Table 1 were created in the calibration sequence of the code.

These variables allow the user to preset every necessary variable to get the most precise measurement that is needed for their type of measurement. These variables are important to note not only for the movement of the device and its communication during said movement, but when using the measured data and when transforming the data the size and location of the measurement area and magnet are needed to be able to solve for them.

**3.2 Arduino PCB**

To connect the motors to the Arduino was a PCB that acted as a hat to the Arduino Uno. The PCB was designed specifically for the movement of this 3D printer. It contains motor controllers for each motor on the 3D-printer, connections for 3 limit switches, and an external power supply for the motors of the 3D-printer. These limit switches are attached to safe points on the 3D printer that give the 3D-printer its internal location to make sure it does not go outside of the bed causing damage to the motors.

The PCB was designed by using a PCB creating software called easy-eda [12]. This program allows for a schematic to be written up of the wiring connections for the Arduino to the motor controllers and limit switches. It is also important to think of long-term use of this device and considering another student or research might use this measurement system again it is important to ensure that there is proper documentation of everything including the wiring. From there the size of the PCB board was designed to be just above the size of the Arduino Uno. And after that the rest of the connections can be placed on the PCB board in a way to not only fit all the motor controllers, but to keep them organized. It is important to note that the external power supply carried a large amount of current that would prove to be fatal to the connecting PC and Arduino UNO if carelessly designed. To prevent this the power supply was designed in a location away from other connections and as close as possible to the motor controllers.

The design for the PCB went through some variations throughout the design process to ensure that all major functions work well, safety of the device and user, and for easy replacement of parts. The process in the original design was simply designed for control of the stepper motors. The stepper motors were to be controlled using a DRV 8825 motor controller. This motor controller was picked not just for simply fitting in the specs of the Creality motor controller, but also because a cheaper motor controller does not have the same protection features like Overcurrent Protection, Thermal Shutdown, VM Undervoltage Lockout, and Fault Condition Indication Pin. This specific motor controller also has a great operating temperature range from -40 to 85 degrees Celsius.

Once decided the motor controllers were first placed on the PCB in locations lined up to easily understand which motor controller controls which axis. To connect the motors to the controllers the pins were connected to a male connection on the PCB next to their respective motor controller. These are the same connectors used for the original 3D-printer, which meant that these motor connections did not need to be cut and if there are any updates to the PCB it can be replaced easily. Next to each individual motor controller was their respective limit switch. In the original design, trying to save space, the motor controllers and limit switches were originally separated, but it was decided that this would prove to cause some problems in the future being confused, so the switches were placed next to their respective motor.

The next step in the process was connecting the motors to power. These 3 stepper motors have a high-power demand meaning a simple USB connection from the Arduinos 5V power supply would not nearly provide enough power to the motors, so an external power supply was used for this process. This external power supply was connected to the PCB board using a screw in style connector. For safety measures there is a switch on the power supply and the wire connecting the PCB to the power supply has female and male connections in case of emergency release from the power supply.

The design for the wiring of the PCB can be seen in figure 7. Once that wiring was confirmed to be safe the next step in a PCB design process is to design the actual PCB board using all the thought processes above locations were determined and finalized in Figure 8. The full soldered and built design can be seen in Figure 9. In the design the separation for motors can switches can be seen. It should be noted that the Power section being in the top left corner is that far away from the rest of the PCB for safety measures. This design on easy-eda was sent to an outside company to be created and then sent to Doshisha for confirmation and further testing.

**4 Measurement System**

**4.1 Lakeshore Tesla-meter**

A Lakeshore Tesla-meter was used for the magnetic field data collection. There are many perks of using this device, like being a rod shape allowed it to be near seamlessly replaced with the extruder from the original 3D printer. On top of that it can measure up to 4 different types of measurements in a single collection. Those being the *x, y*, *z* axis respectively and the average strength of the field at that given point. The orientation of the Measurement for the *x, y*, *z* orientation can be seen in Figure 6. With its conveniently pen shaped end the Lakeshore Tesla-meter while being attached to the 3D printer’s extruder measurement end the data is sent to its main device in live time. The company Lakeshore has its own python library for this device to allow more automatic measurements, timing, and other functions that allow this data to be collected smoothly.

**4.1.1 3D Printed Mount**

To replace the 3D filament extruding attachment for the Lakeshore Tesla-meter it seemed like a simple process, mainly due to the fact the enders 3 3D printer was a very popular inexpensive 3D printer meant that there were a lot of open-source designs for different types of attachments like pens, pencils, etc. Originally believing it would be a simple switch, an open-source design was used and installed on the machine. Data measurements were found to have a lot of unwanted noise and unexplainable changes of measurement strength throughout the process [9]. This was due to the open-source pen design and the Tesla-meter being incompatible sizes even when originally thought to be properly mounted there were still too many inconsistencies that led to this design proving to be inaccurate and unusable.

The solution to this was to design and print a custom 3D mounting device for the Lakeshore Tesla-meter. Doing so would ensure proper safety of the device and accurate measurements throughout all the data measurement process. The mount was designed in Autodesk Fusion 360 to the shape of the original 3D filament extruder mounting holes and then extended to then fit the Tesla-meter and leave extended room long enough to support its pen shape throughout the whole mount. The longer design meant that it could safely handle unwanted pulls to the device even when mounted in place. This would ensure safe measurements and safety of the device during and outside of the measurements. The final design of the device can be seen in a design at Figure 1. It being attached to the 3D printer can be seen at Figure 2. As seen from Figure’s 1 and 2, the design allows for a comfortable fit onto the 3D printer and for the Tesla-meter. Figure 3 contains a different angle of the attachment on the 3D printer

**4.2 Python Code**

With the library for taking measurements of the Lakeshore Tesla-meter being written in python [11], it was decided that it would prove to be easiest to write the code controlling all actions of the measurement process to also be in python. The measurement code would do all functions needed to accurately measure and write the data. This includes external writing to an excel file. Before writing the code, what would need to be done would need to be decided first. With the movement code needing to be written separately in Arduino IDE, communication, timing, and set up would all be done in python.

For set up all device’s connections would first need to be establish and if not find the issue as to why not. This meant establishing a connection to the Lakeshore Tesla-meter and the Arduino controlling the 3D printer. With these devices connected the code could proceed. It is important to take this step first because one, nothing could be accomplished without a proper connection, but two improper connections to one or another could mean false data would be collected. The Arduino and Lakeshore Tesla-meter are connected to the main PC running python code via Serial connections. It is important to note that only one program can access these serial connections at a time. When troubleshooting movement, it is very nice to have the Arduino print its errors to the internal serial connection monitor, but this cannot be done when running the python data collection program so any errors from this step forward on the movement side would need to be solved by communicating its issues to the main PC. At first the Arduino was supposed to have only one way communication only receiving commands on directions, calibration, etc., but this proved to not be ideal, since not knowing what was happening inside of the Arduino would mean if there were any errors in the movement side of the measurements that were not physically visible there would be no way to know if they had occurred.

The next step is to do any preprocess setups like sending a signal to the Arduino printer to calibrate itself and to build a csv file and separate where the data collected will be written. The csv file records the current date and time of the collection. The calibration process of the 3D printer can take a little bit of time so the code accounts for this and waits and during that time updates from the 3D printer are sent back to the main PC and status of the calibration and of the movement of the 3D printer can be checked during this time. The code then goes through a loop for the number of rows and columns of the measurement process and for each loop sends a signal to the 3D printer to move to each point and once at that point the PC takes 5 measurements, and the next loop begins. These 5 measurements are taken to after averaging them to ensure accurate results throughout the whole process at each point. The commands sent to the Arduino are labeled with which number in the row they are collecting the column they are at; the Arduino takes this line of command and breaks it apart to only receive the number value of the row they are at. This number signifies not only its position but where in the row it is so that at the end of the row it can reset itself into position 1 of the new column. Sometimes these positions, depending on the user settings, can exceed ten, this can cause some errors if proper caution is not taken beforehand. Signals sent to the Arduino must be sent via string, when parsing the string the Arduino must be careful to receive the correct instructions. To ensure that it has no errors the data is parsed using the position of the string for the numbered part of the instructions so when sending data, it is always sent in 3 digits. For example, 001, 010, 100 will be sent for the numbered part of the instructions for one ten and one hundred respectively. A visual of the area of the measurement and an example of how it would occur can be found at Figure 4 as the it is in person and in figure 5 as it was concepted.

Between switching to a new column, the code gives time for the 3D printer to reset itself and when it does the Arduino then sends a signal saying it is ready to collect. This process continues for a total of rows times columns collecting a total of that number times 5 for the averaging measurements. The csv file containing all the data is completed and ready to be exported if the data collection occurred without an issue. For a simple example diagram of how the measurement process measures a row and then switches to a new column see figure 10.

Throughout the measurement process the data returned from the Arduino, the status of the measurements from the Tesla-meter, and the status of the program are all outputted in the output of the python code for ease of use. Sometimes where this has come in handy as previously stated is that the Arduino can only communicate using its serial port through one port at a time meaning that any outputs need to be received by the python code and then outputted from there. This at some points can mean too much data is being sent all at once considering there are 900 measurement points, so if an issue occurs it is difficult to find it if only one, the Tesla-meter or the 3D printer, fail.

To make errors and smaller issues easy to find the code will, one stop if it loses communication or connection to either the Tesla-meter or the 3D printer outputting the location of the stop, and two collect every measurement instance output and organize them together to allow easy readability of the condition of the measurement system. In early stages of the system where timing issues occurred frequently, before establishing a proper two-way communication system, these codes were detrimental to safety of the measurement process, because relying only on one-way commands and timing left a lot of room for error.

Even after many issues have been fixed these status updates are important to have when the measurement process occurs. Now the current research takes 900 measurements over a 100 mm by 100 mm area, with 5 measurements per each 900 measurement points the process can take up to two hours. Throughout the process if a small mistake occurs it would make the whole measurement process pointless so being able to track the status throughout the measurement process is very important.

**5 Magnetization Estimation in a Permanent Magnet**

**5.1 Data Manipulation**

Once the data is collected it is checked over to make sure the collected data matches the expected data. For example, in the case of 50 measurements per row and 50 measurements per column there should be a total of 2500 times 5, so 7500 measurements per *x, y, z,* and absolute value measurement respectively. This accounts for each direction and the averaging values. If this is not the case, then somewhere the data collection went wrong and needs to be redone.

This data is collected onto a CSV file which is used and exported into matlab to separate each direction and average. This being the data for the magnetic field it is best to label each direction ***B*** followed by its respective direction, that being ***Bx****,* ***By***, ***Bz****,* and ***B***abs. The Code considers the measurements per rows and columns and using that creates an inside and outside loop to properly parse the data. The inner for the rows and outside for the columns. The loops all do the same function of averaging the 5 measurements to one value and input that to each location of the loop to its corresponding position for each direction of ***B***.

**5.2 Magnetic field to Magnetization**

**5.2.1 Biot-Savart Law**

In this research permanent magnets were used and measured. Doing so ensures a constant magnetic field and magnetization inside of the magnet. Thus, it is acceptable to use Biot-Savart law. Using research from a collection of magnetic research journals [4], equation from this journal contained methods to obtain the magnetization of said magnet from a given magnetic field.

(1)

From Biot-Savart law *I* can be found and transformed into the following equation.

(2)

Though in a permanent magnet such as what this research is conducting there is no current, using this relationship from Biot-Savart law allows us to solve for ***M*** from the magnetic field ***B***. In this case, where there is an *I*x, *I*y, and *I*z. There are two sets of 3D locational points that refer to the location of the magnetic field and the magnetization relative location at that relative location, p. In this research we can used the measurements’ location points as the locations for the 3D points , , and . While the points , , and refer to the location of the magnetization [4]. In this research these points must be consisting of the area of the magnet to perform this transformation. Looking at figure 11 shows the correlation between the locational points of the magnetic field and the magnetization. These points are arbitrary when choosing a location p, which is the measurement point. This point varies throughout the measurement process and not only that, but this point p must create a locational matrix for every instance of the magnetization creating a large matrix of locational matrices for the size of the magnetic field and magnetization matrices.

(3)

Seen from equation 3, the *x*, *y*, *z* direction values of ***B*** and ***M*** can be found from one another once the inner matrix is solved. Though looking complicated the inner values of this matrix can all be solved for using equation 2. It is important to note that there appear to be repeats in some of the matrix values this is since in this case and will result in the same value so for the sake of efficiency one is simply replacing the other.

(4-a)

(4-b)

(4-c)

The equation’s in 4a, 4b, and 4c represent how the values would be found from equation 2 by taking the partial derivative of each respective variable. By doing so the data of the magnetic field taken at every point is correlated with its locational point, *p*, and then built into a matrix to solve for the Magnetization, ***M***. It is very important to note that to solve for Magnetization, ***M*** the inverse of the inner matrix will need to be solved. To make things simpler for this explanation the equation will be read as ***B*** = ***KM*** where matrix ***K*** contains 1/(4π) and the matrix with locational values [5].

**5.2.2 TSVD for Matrix Solving**

Singular Value decomposition is used for matrix processing since the pseudo-inverse can be very easily solved with this method. The matrix ***K***cannot be rigorously transformed into an inverse matrix so the pseudo-inverse of ***K***must be solved for using SVD [5][3]. Doing so would turn into ***M***=***K***\* ***B***, where ***K***\* is the pseudo-inverse of matrix ***K***. The process to do this takes the matrix ***K*** and decomposes it into the following expression

***K = USVT***  (5)

In equation 5, the variables **USV** contains the different vector matrices and singular values of matrix ***K***. ***K*** is an *m* by *n* matrix referring to the sizes of matrix ***B****x*, ***B****y*, ***Bz***, ***Mx***, ***My***, and ***Mz****.* Matrix ***U*** is an *m* by *m* matrix, ***S*** is an *m* by *n* matrix, and ***V*** is an *n* by *n* matrix. Finding the pseudo-inverse turns ***S***-1 to an *n* by *m* sized matrix.

During the measurement process it is comparing for little bits of noise to occur, especially with these smaller types of measurements. This could be due to interference of the motors themselves moving the 3D printer, high powered electronics in the Laboratory, etc. This noise could in some cases make the given data look very wrong so truncating the data in the SVD process helps remove some of this noise.

Using Truncated Singular Value Decomposition mainly uses the matrix***S*** when we used SVD on our matrix ***K***. This matrix has a rank *r* containing singular values. The idea is to truncate the matrix ***K*** by reducing the rank *r*. This can be done knowing the fact of the singular values. In the singular value matrix ***S***. The singular values are contained in a diagonal matrix the size of *m* by *n*. From the top down the diagonal singular values slowly reduce in size. It can be found that by selecting a singular value size and removing any singular values below that threshold then resizing the rank to the new rank of matrix ***S*** with the singular values removed can do this. Then in the same process as normal SVD using these new reduced matrices to solve for our original matrix ***K*** and its pseudo inverse ***K***\*.

. (6)

Figure displays the Singular Values of the ***S*** matrix part of the SVD for the inverse of Matrix ***K***. This size is due to the magnetic matrix rank and the results cannot be higher than that given rank. On the y axis is the normalized value, *σ*k. The normalized value threshold is what is going to be reduced using εn, which will determine how many singular values, *σ*k, are to be rid of. This is the most important part of the truncation process as εn directly affects the amount of data truncated throughout this process. Picking numbers to truncate at random may prove to be difficult in this case where there is a large rank. Looking at the singular value graph, Figure 12, can help determine the right number of singular values to remove. As seen in Figure 12, the values seem to end at near 10^-2, measured from graph and data the normalized values with the most important are at singular value 900 where the normalized value is 0.0146. This means that any values less than said value will have no effect and any magnetic field using those normalized values will be essentially the same as Figure 14, the original measurement.

Using the graph to find any major changes the other normalized values of importance are at singular values, *σ*k, 0.02, 0.05, 0.1, 0.2, and 0.5 due to sharp changes. The graph of Figure 12 is created by visualizing the values of the ***S*** of ***USV*** matrices from the SVD of the matrix ***K***.

**5.3 Practical Application**

For the transformations Matlab was used not only for parsing the data, but also to visualize the original magnetic field ***B*** and to solve for magnetization, ***M***. The process involved many large matrices and data exchanging, which is another reason why using SVD is a great option, due to its efficient way of doing these types of calculations.

The code involved first setting the variables for the measurements and to the magnet. This means establishing the size of the measurement area, its distance to the magnet, the magnet’s size, and its location relative to the measured area. This is important for the previously stated method of developing matrix ***K***. The magnet’s magnetization matrix size does not need to be the same size as the measurement matrix and is chosen in these steps.

For practical reasons an example magnet and situation will be used for explanation. In this scenario a 100 mm by 100 mm measurement area is used for 50 by 50 measurements. This means 2 mm spacing between measurement locations on the x and y axis. The magnet used will be placed at the exact center of the measurement table and is 20 cm by 60cm. The magnet used in this research can be found at table 2 in the appendix, for a list of its properties. Creating a matrix size with similar dimensions to the size the magnetization matrix was chosen to be 10 rows by 30 columns. This also does mean that the distance between relative magnetization points is also 2 mm. This does not need to be the case and any size and distance between points can be chosen, but for this scenario was the case.

In this scenario the size of matrix for***B***and ***M*** would be 7500 and 900 referring to the 3 directions *x*, *y*, *z* and the size of the magnetic field matrix, 50 by 50, and the magnetization matrix size, 10 by 30. A loop is construction to go through each point of the measurements for ***B*** use the previous equation’s for solving the partial derivatives in equation 2 to build a matrix ***K*** for each location of the decided magnetization, ***M*** points. This creates a 7500 by 900 matrix ***K***.

***K\****

(7)

This pseudo-inverse of matrix ***K*** needs to be solved for to solve for magnetization, ***M***. Matlab has SVD functions built into so using SVD on matrix ***K*** gets three separate matrixes stated before. These matrices ***U***, ***S***, and ***V*** are used and using previous equation 4, solves the pseudo-inverse. From getting the pseudo-inverse a singular value is selected to reduce the rank of the SVD matrix ***S***. From here the new matrix is used to replace the original pseudo inverse matrix ***K***\*. This new pseudo-inverse***K***\* is multiplied by matrix ***B*** to solve for ***M*.**

As seen in equation 7, when the singular values *σ*k are summated the smaller the value the more unstable the data can become. This is another reason why truncating is a very important process not just to remove noise but, in this case, since the inverse cannot be directly solved for find thing the pseudo-inverse using SVD is the only option and to make that process more stable truncation is used. It is important to note when this method was used in previous research using a similar method for finding the current distribution density of a PV panel without using truncation produced very poor results versus using truncation [7].

As the singular value *σ*k, decreases so does the rank, r, of the matrix to be r’. The singular values being truncated changes the overall rank size, r’, of the matrix so the new rank size must be smaller than the original rank, r. Truncating too much and reducing the rank will make the whole process unstable so it is important to know that there is a balance between truncating too much and too little, that being adding all the singular values, *σ*k, will result in instability, and reducing the rank size, r’, will also create instability, so there is said to be an ideal value somewhere.

**6 Results**

**6.1 Simulated Results**

To comparing data an ideal result, a simulated version of the magnetization of the magnet of the same size was simulated. This was then transformed using the same process in reverse of finding ***M*** to find ***B*** in this case. To accomplish this an ideal magnet would need to be constructed. Matlab was used for this process since all of the calculations were already being done in Matlab it would make transforming the simulated results of magnetization to the magnetic field. For this simulation it was to get an ideal shape of the magnetic field.

One important way to see if the SVD matrix solving was successful is to reverse the process and obtain the magnetic field ***B*** again from the solved magnetization, ***M***. This result is the ideal magnetic field shape and removing noise from the measured magnetic field should look as close to the original measured magnetic field as possible to confirm the magnetization solving was successful.

Continuing with the simulation process, for the purpose of comparing the results to the previous example a constructed magnetization area of 10 by 30 was created. The same ***K*** and ***K***\* matrices previously solved for can be used due to the fact we are transforming the same size magnet, simulated or not, to a 100 mm by 100 mm magnetic field area for a direct comparison. The magnitude of the magnetization in this 10 by 30 grid would be zero for the *x* and *y* values of ***M*** and be 1 for the z directional value. This is because in an ideal magnet there would be no directional *x* and *y* magnetization magnitude and using 1 Tesla would make solving very easy to compare results to.

The 10 by 30 matrix for magnetization, ***M***, is then multiplied by the original matrix ***K*** since that matrix contains the locations and size of the measurement area and magnet, so means it can be used without issue. The result would be magnetic field ***B*** and contains 3 dimensional values that can be compared to later. The 3D graphs of the magnetic field created from this simulation can be found at Figure 13.

**6.2 Magnetic Field Results**

Using the previous methods of collecting the data for the magnetic field and Matlab for visualizing the magnetic field the measured magnetic field was created seen in Figure 15 for the first measured magnet that is 20 cm by 60 cm. Comparing this result to the simulated magnetic field in Figure 13 the general shape and strength of the magnetic field is very similar to each other. This is a good start to understand that the measured magnetic field is accurate with no major flaws. Though there is a little bit of noise in the measured magnetic field, using TSVD should help clean up and erase this noise.

The noise in Figure 15 can be seen around the surrounding measuring area of the magnet. There is a little ripples and inconsistencies that prove that there is noise there. Looking directly at the magnetization or the area at the magnet can in some cases can be difficult to see so in the cases of visualizing the magnetic field this is a very simple way to tell there is noise in the data.

**6.3 Magnetization Results**

When using the previously shown methods to get the magnetization of the measured magnet used in this research there are variable options for choosing a singular value that will remove the ideal amount of noise in this scenario. TSVD is a new method, and it is considered extremely difficult to choose this value. For initial testing of this value, it is not only important to get a clean look at magnetization, but to use said magnetization and get the magnetic field using the same method to get a visual magnetic field as close to the simulated field as possible. To do this a wide range of values were initially tested.

With the size of the magnets rank being 900 using the smallest singular value, *σ*k, found was found using Figure 12 being 0.0146, any value lower than this number being used as a normalized value for truncation would be pointless. The truncation method works by allowing numbers greater than the normalized value and not reducing the rank of the matrix ***S*** at all, so a number lower than that would simply let every normalized value through not changing the rank of matrix ***S***. For these tests, seen starting at Figures 9 and 10, the initial normalized value of εn, the cut off value, was chosen to be 0.02. This value was chosen when looking at Figure 12 and looking at the smallest normalized value of 0.0146 and with both in mind determine the starting point at 0.02. As previously stated, other points of interest along the curve of the singular value numbers at Figure 12, were 0.05, 0.1, 0.2, and 0.5. The plan was to visualize all the values starting from the smallest values up. This was because on the extreme ends removing too much data caused the returned magnetic field to become a totally different shape and the removed data creates gaps in the visualized graph making the correct data look like it was noise itself.

This method of comparing the magnetic field shape is very important, but another thing to consider is what the magnetization field would look like from the simulated magnetic field being put through the TSVD process as if it was just a standard measurement done by the system. Though expecting the magnetism throughout the magnet by this method to be very consistent to the values initially chosen for the magnetism of the simulated magnet, it is important to see the results of doing so for a proper comparison. Results for this seen at Figure 10 can be seen to yes have a similar magnetism to the initialized value. Looking at this also helps to determine that ideally the magnetization in the *z* direction is very important and ideally should be flat and consistent. Looking at the results of the magnetism from the measurement without any truncation, Figure 16, there can be some consistency being seen that hopefully through truncation can be flattened to look like the results from the magnetism from the simulated magnetic field in the *z* direction.

Starting from εn=0.02 at Figure 17 and Figure 18 for the magnetism and magnetic field, the graph, having only a small amount of data removed looks very similar to the original graph, but with a slight amount of the noise removed. At first glance it could be mistaken for the simulated magnetism to magnetic field graph. This graph had a rank of 892 from the original 900 rank. Truly only a slight amount of data was reduced.

Next was the value εn=0.05, in this process the new rank was 854 which is removing a substantial amount more data than previous. The graphs can be found in Figure 19 and Figure 20. Originally looking at the magnetic field graph. The results looked ideal, due to the magnetic field graphs seem to have all noise removed from the graph and keep a good shape it seemed like a good point. Before determining this to be the end the next step was to continue and compare.

Continuing to εn=0.1 and εn=0.2, the graphs can be seen from Figure 21-24. The magnetic field graphs initially looked to be clean with little to no issue with the noise, but only comparing the magnetic field graph has its downsides. Looking at the magnetization at these normalized values the magnetism seemed to be very ugly and with a lot of unnecessary noise.

With this knowledge of the magnetism, Figures 17 and 19 were reviewed again. In the simulated testing the magnetism being used was (0, 0, 1) Tesla for the *z, y, z*, directions respectively. In the idea situation the *z* direction of the magnetism should be consistent at least towards the center. Reviewing Figure 23(c), the magnetism can be seen to have an unnecessary dip meaning that most likely too much data is being removed. Continuing to 0.5 was determined to be a pointless cause and a new value was determined from the first two normalized values being in between 0.02 and 0.05, that being 0.035.

The Figures 25 and 26 display the magnetic field and magnetism with a normalized value of εn=0.035. Here the final rank is 886 with only removing a little bit more data than the case of εn=0.02. Looking at the magnetism at Figure 25(c) the magnetism looks ideal being fairly the same value and smooth near the center. This was a good starting point meaning that most likely too much data was not removed. The next step was looking at the magnetic field figures at Figure 26. The surrounding area had no noise and had a great shape meaning that this normalized value was most likely very near to the ideal normalized value.

Though visual keys are important to understanding the results it is important to have a number to compare the difference between the results. For reference the magnetic field was found again from the magnetization with no truncation were compared using the original magnetic field measurement results to check for differences and found a though visual similarity between the two there was a 1.63% difference between the two results, proving that the process was accurate throughout the SVD method. Next up is comparing to the simulated results to the original measurement with no truncation and the results with truncation at εn=0.035. The main issue is that can be seen from the simulated magnetic field results is that though sharing a similar shape, since it is simulated, the results will still have a much different value. The results found that compared to the original results the magnetic field at εn=0.035 was 64.97% more like the simulated results compared to the measured magnetic field. It is important to note there was a 3.15% difference between the measured magnetic field and the magnetic field from the magnetization at εn=0.035, though only a small difference between the two, the large difference between the simulated magnetic field and the measured magnetic field made the comparison to be larger than anticipated.

**7 Conclusions**

The research has gone through many interactions and going back there are likely methods to have efficiently get to this point at the research, but the steps involved carry their own importance to the overall growth of the research and finding the next steps to this research.

Containing two important research topics in this research, being the measurement system and the transformation of magnetic field B to magnetism **M**, each individually are not bound to each other and can be used in a wide range of applications on their own. For example, the measurement system can easily be converted for measuring the magnetic field of electronics, or thinking bigger, another sensor could be attached to the sensor location to collect any type of data. The method of transforming magnetic field B to magnetism M is also very important and can easily be used by itself in the reverse to determine ideal magnetic fields of given magnets.

This research overall has a wide range of information and equations that are all individually important to getting the result. For this research the extent was getting as near as possible to the ideal normalized value and being able to visualize it. For continuing from this point, it would prove to be important to find the exact normalized value for the TSVD process. With AI and Machine Learning becoming more and more popular in the past century having methods like TSVD are very important for not only removing noise but data reduction when working with large matrices [6]. These methods are applied to remove as much data as possible within a certain threshold to keep the bulk of the most important data.

**References**

1. N. Nakai, Y. Takahashi, K. Fujiwara, and H. Ohashi, “Estimation of Magnetization Distribution in a Permanent Magnet using Genetic Algorithm,” *Proc. 17th Int. Symp. Electromagn. Fields Mechtron. Elect. Electron. Eng. (ISEF)*, Paper JP171, 2015.
2. L. Arbenz, O. Chadebec, C. Espanet, Y. Rtimi, and G. Cauffet, “Characterization of Permanent Magnet Magnetization,” *IEEE Trans. Magn.*, vol. 53, no. 11, Art. no. 8109504, 2017.
3. N. Nakamura, Y. Okamoto, K. Osanai, S. Doi, T. Aoki, and K. Okazaki, “Magnetization Estimation Method for Permanent Magnet Based on Mathematical Programming Combined with Sigmoid Function,” *IEEE Trans. Magn.*, vol. 58, no. 9, Art. no. 6000704, 2022.
4. S. Wakao, H. Igarashi, K. Fujiwara, A. Kameari, “Useful Formulas of Analytical Integration in Electromagnetic Field Computations (Part 1)” *Abstract Contents of Technical Report*, No.1043, 2006.
5. N. Nakamura, Y. Okamoto, K. Osanai, S. Doi, T. Aoki, and K. Okazaki, “Nondestructive Estimation of Magnetization Distribution in Permanent Magnet Using Quasi-Newton Method Based on 2-D Fourier Series” *IEEE Trans. Magn.*, vol. 56, no. 1, Art. no. 6000305, 2020.
6. M. Yeh “An Efficient, Tolerance-Based Algorithm for the Truncated SVD” *UC Berkeley Theses and Dissertations*, 2022.
7. H. Nishimi, Y. Ohnishi, Y. Takahashi, K. Fujiwara, “Visualizaition of Current Density Distribution in Photovoltaic Modules Using Truncated Singular Value Decomposition Based on Measurement of Surrounding Magnetic Flux Densities” *Doshisha University*, 2021.
8. D. Lausch, M. Patzold, M. Rudolph, C.-M. Lin, J. Fröbel, and K. Kaufmann, “Magnetic Field Imaging (MFI) of Solar Modules”, *Proc. of 35th EUPVSEC*, pp. 1060-1064, 2018.
9. H. Igarashi, T. Honma, A. Kost, “Inverse Inference of Magnetization Distribution in Cylindrical Permanent Magnets”, *IEEE Transactions On Magnets,* Vol. 36, No. 4, 2000.
10. PyCharm, JetBrains, 2024. [Online]. Available: https://www.jetbrains.com/pycharm/
11. LakeShore Python Library, Lake Shore Cryotronics, 2024. [Online]. Available: https://github.com/lakeshorecryotronics
12. PCB Design Software, Easy Eda, JetBrains, 2024. [Online]. Available: https://easyeda.com/
13. Arduino. 2024. Arduino IDE [Computer Software], Version 2.2.2. Available: https://www.arduino.cc/
14. M. McCauley, P. Wasp, “AccellStepper”, Version 1.64, Available: https://github.com/waspinator/AccelStepper, 2024

**Appendix A. LIST OF TABLES AND FIGURES**

|  |  |  |
| --- | --- | --- |
|  |  | Page |
| Figure 1 | 3D Model of Lakeshore Tesla-meter mount | 22 |
| Figure 2 | Tesla-meter attachment | 22 |
| Figure 3 | Magnetic field measurement system (real image) | 23 |
| Figure 4 | Measurement system visual | 23 |
| Figure 5 | Magnetic field measurement system (drawing) | 24 |
| Figure 6 | Tesla-meter magnetic field orientation | 24 |
| Figure 7 | Wiring schematic of 3D printer Arduino UNO PCB | 25 |
| Figure 8 | Arduino UNO PCB | 25 |
| Figure 9 | PCB completed | 26 |
| Figure 10 | Data collection order | 26 |
| Figure 11 | Magnetization and magnetic field locational diagram | 27 |
| Figure 12 | Singular value graph | 27 |
| Figure 13 | Simulated magnetic field | 28 |
| Figure 14 | Magnetization from simulated magnetic field | 29 |
| Figure 15 | Measured magnetic field | 30 |
| Figure 16 | Magnetization from measured magnetic field | 31 |
| Figure 17 | Magnetization for εn =0.02 | 32 |
| Figure 18 | Magnetic Field for εn =0.02 | 33 |
| Figure 19 | Magnetization for εn =0.05 | 34 |
| Figure 20 | Magnetic Field for εn =0.05 | 35 |
| Figure 21 | Magnetization for εn =0.1 | 36 |
| Figure 22 | Magnetic Field for εn =0.1 | 37 |
| Figure 23 | Magnetization for εn =0.2 | 38 |
| Figure 24 | Magnetic Field for εn =0.2 | 39 |
| Figure 25 | Magnetization for εn =0.035 | 40 |
| Figure 26 | Magnetic Field for εn =0.035 | 41 |
| Table 1 | Table 1 Arduino variables | 42 |
| Table 2 | Measured magnet properties | 42 |
|  |  |  |
|  |  |  |

A blueprint of a machine

Description automatically generated

Figure 1 3D Model of Lakeshore Tesla-meter mount

A machine on a table

Description automatically generated  
Figure 2 Tesla-meter attachment

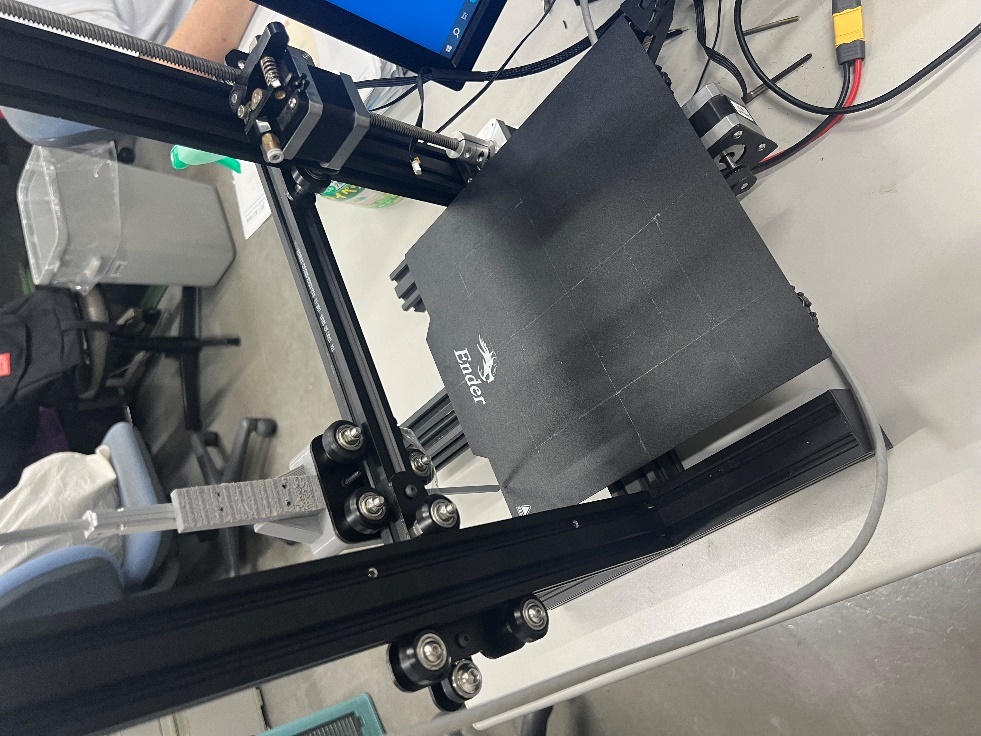


Figure 3 Magnetic field measurement system (real image)

A blue and black object with a square object in the middle

Description automatically generated with medium confidence

Figure 4 Magnetic field measurement system (drawing)

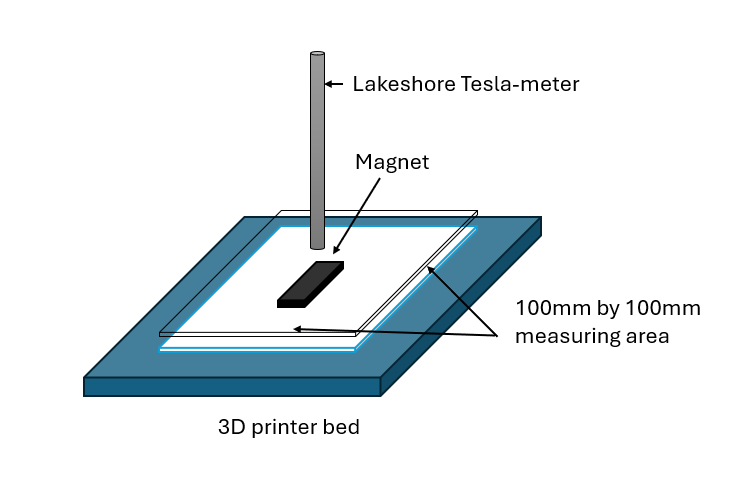


Figure 5 Measurement system visual

A diagram of a rectangular object

Description automatically generated

Figure 6 Tesla-meter magnetic field orientation

A diagram of a computer

Description automatically generated

Figure 7 Wiring schematic of 3D printer Arduino UNO PCB

A green circuit board with yellow dots and white dots

Description automatically generated

Figure 8 Arduino UNO PCB

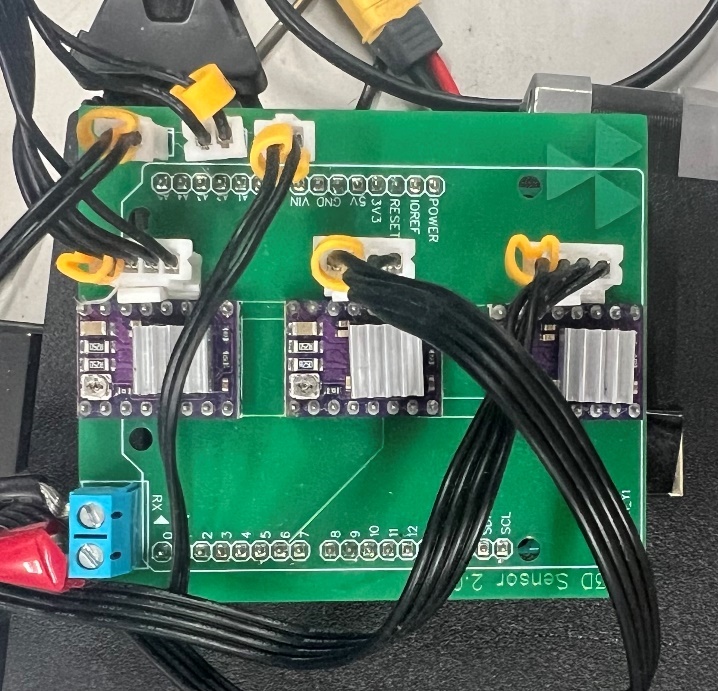


Figure 9 PCB completed

A black rectangle with arrows pointing to a black rectangle

Description automatically generated

Figure 10 Data collection order

A diagram of a magnetic field

Description automatically generated

Figure 11 Magnetization and magnetic field locational diagram

A graph with a line

Description automatically generated

Figure 12 Singular value graph

A graph of a magnetic field

Description automatically generated

(a) *B*x

A graph of a magnetic field

Description automatically generated

(b) *B*y

A graph of a magnetic field

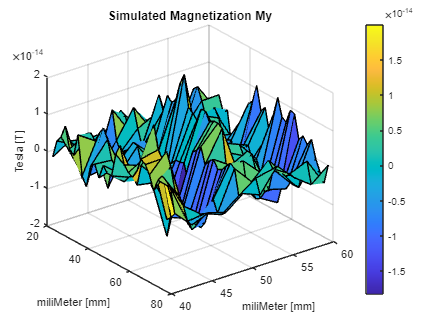
Description automatically generated

(c) *B*z

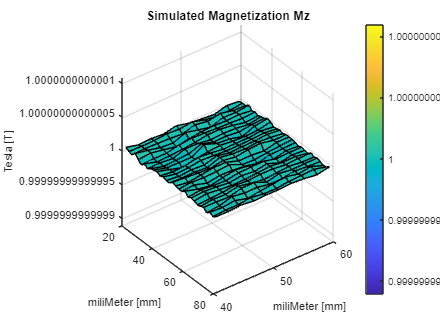
Figure 13 Simulated magnetic field



(a) *M*x



(b) *M*y



(c) *M*z

Figure 14 Magnetization from Simulated Magnetic Field

A graph of a magnetic field

Description automatically generated

(a) *B*x

A graph of a magnetic field

Description automatically generated

(b) *B*y

A graph of a magnetic field

Description automatically generated

(c) *B*z

Figure 15 Measured Magnetic Field

A graph of a graph showing a curve

Description automatically generated with medium confidence

(a) *M*x

A graph of a graph showing a colorful curve

Description automatically generated with medium confidence

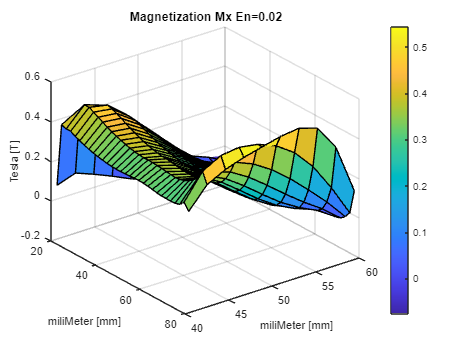
(b) *M*y

A graph of a graph with a rainbow colored curve

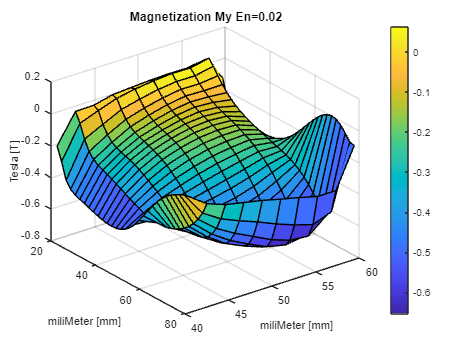
Description automatically generated with medium confidence

(c) *M*z

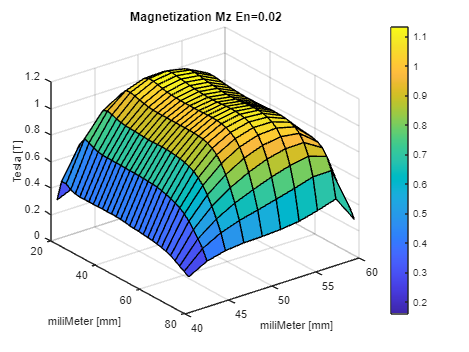
Figure 16 Magnetization from Measurement



(a) *M*x

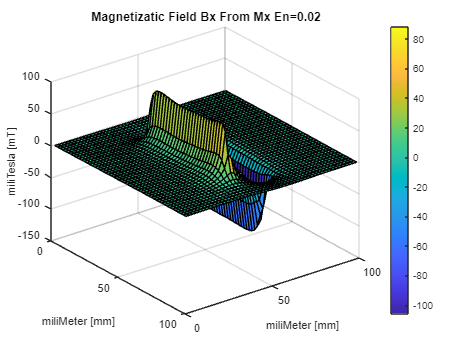


(b) *M*y

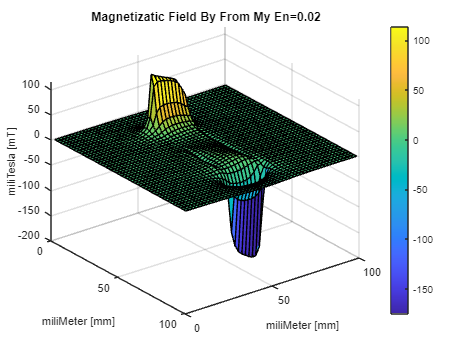


(c) *M*z

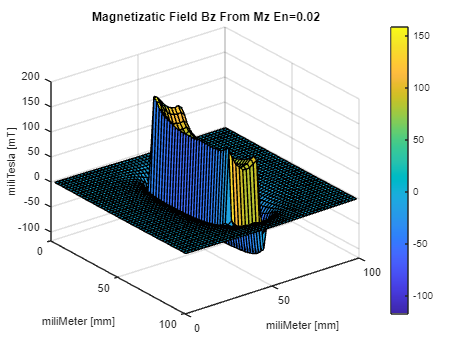
Figure 17 Magnetization for εn=0.02



(a) *B*x



(b) *B*y



(c) *B*z

Figure 18 Magnetic Field from Magnetization for εn=0.02

A graph of a graph

Description automatically generated

(a) *M*x

A graph of a graph

Description automatically generated

(b) *M*y

A graph of a graph showing a curve

Description automatically generated with medium confidence

(c) *M*z

Figure 19 Magnetization for εn=0.05

A graph of a graph showing a field

Description automatically generated with medium confidence

(a) *B*x

A graph of a field

Description automatically generated

(b) *B*y

A graph of a graph showing a graph of a graph

Description automatically generated with medium confidence

(c) *B*z

Figure 20 Magnetic Field from Magnetization for εn=0.05

A graph of a graph

Description automatically generated

(a) *M*x

A colorful graph of a graph

Description automatically generated with medium confidence

(b) *M*y

A graph of a graph showing a curve

Description automatically generated with medium confidence

(c) *M*z

Figure 21 Magnetization for εn=0.1

A graph of a graph showing a graph of a field

Description automatically generated with medium confidence

(a) *B*x

A graph of a magnetic field

Description automatically generated

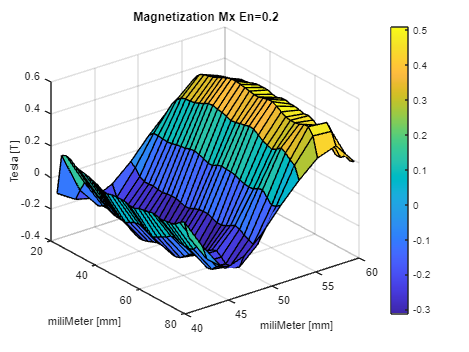
(b) *B*y

A graph of a graph showing a graph of a field

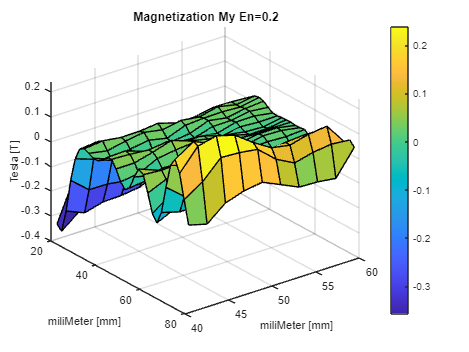
Description automatically generated

(c) *B*z

Figure 22 Magnetic Field from Magnetization for εn=0.1



(a) *M*x

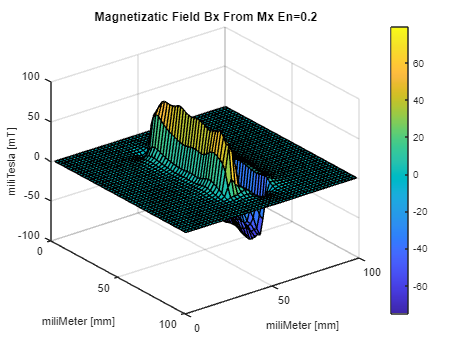


(b) *M*y

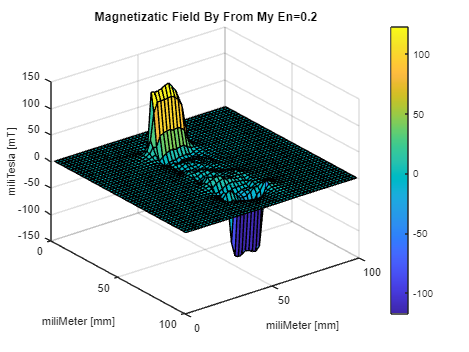


(c) *M*z

Figure 23 Magnetization for εn=0.2



(a) *B*x

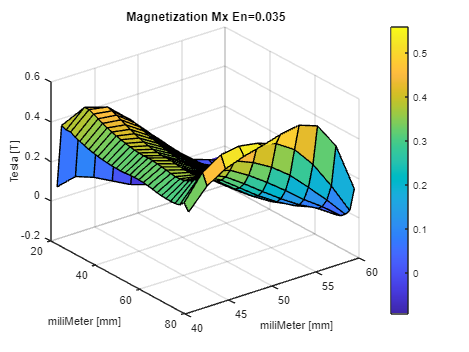


(b) *B*y

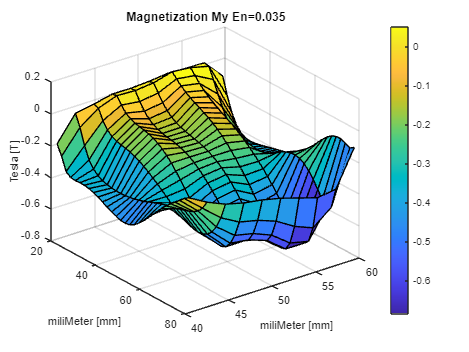


(c) *B*z

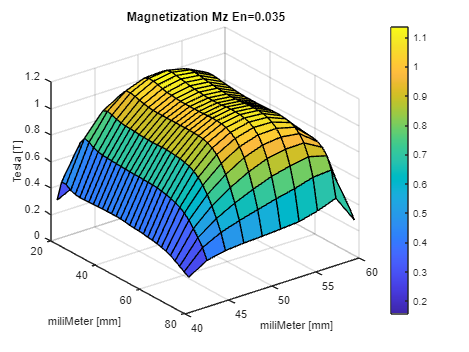
Figure 24 Magnetic Field from Magnetization for εn=0.2



(a) *M*x

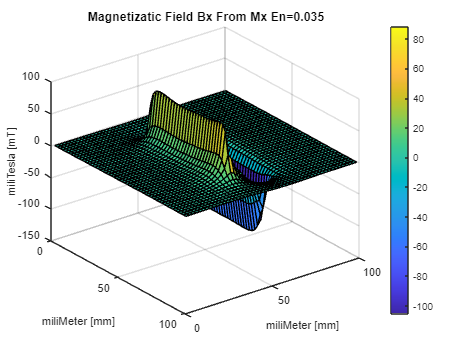


(b) *M*y

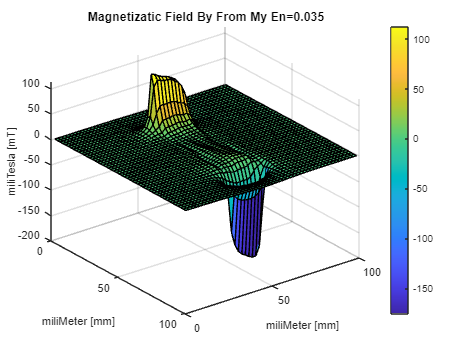


(c) *M*z

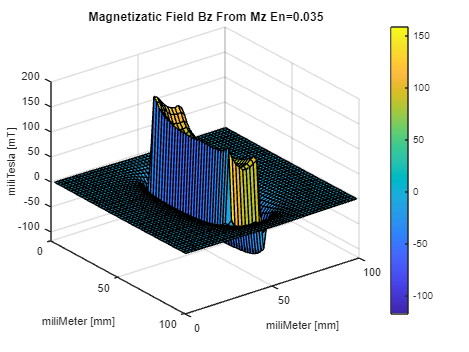
Figure 25 Magnetization for εn=0.035



(a) *B*x



(b) *B*y



(c) *B*z

Figure 26 Magnetic Field from Magnetization for εn=0.035

Table 1 Arduino variables

|  |  |
| --- | --- |
| x\_offset | distance from x=0 of the printer bed |
| x\_length | length of measurements in the y direction |
| x\_measurements | number of measurements in the x direction (rows) |
| y\_offset | distance from y=0 of the printer bed |
| y\_length | length of measurements in the y direction |
| y\_measurements | number of measurements in the y direction (columns) |

Table 2 Measured magnet properties

|  |  |  |
| --- | --- | --- |
| Permanent magnet | Type | Nd2Fe14B |
| Magnetization direction | Parallel |
| Magnetism [T] | 1.225 T |
| Relative permeability | 1.05 |
| Conductivity [S/m] | 6.944 105 |