POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

COMPUTATIONAL ELECTROMAGNETICS - 059445

Biot-Savart law implementation for magnetic field calculation of an 8-figure coil: line elements vs. dipoles

Author: Irene Caridi

Professor: Luca Di Rienzo

Academic year: 2023-2024

---

1. Introduction

Transcranial Magnetic Stimulation is based on the principle of electromagnetic induction, allowing the stimulation of neuronal activity in specific brain areas. This is achieved using a coil, often an 8-figure coil, that stimulates specific parts of the brain. Simulating this stimulation offers numerous advantages, including an enhanced understanding of brain functions, potential therapeutic applications for neurological and psychiatric conditions, and non-invasive mapping of brain activity.
Starting from SimNIBS [1], free and open-source software for simulating non-invasive brain stimulation, this study centers on modeling an 8-figure coil. The coil can be represented either as a combination of line elements or a series of dipole points. The project aims to study the accuracy of the magnetic field calculated with dipoles, taken as a reference to the magnetic field calculated with linear elements.
To evaluate the dipole approximation, the magnetic field and its components are computed using the Biot-Savart law for various configurations and distributions of dipoles. Then, the mean squared error, mean absolute error and relative error are calculated by comparing these configurations and the magnetic field obtained using the Biot-Savart law with line elements.

2. SimNIBS implementation

2.1. SimNIBS software

SimNIBS allows for realistic calculations of the electric field induced by transcranial magnetic stimulation (TMS) and transcranial electric stimulation (TES). The software combines various techniques, including automatic segmentation of MRI images, generation of tetrahedral meshes of the brain, and calculation of electric and magnetic fields using advanced numerical methods.

2.2. Dipoles coil

The coil reconstruction approach generates accurate vector potential distributions for arbitrary winding geometries and this still works when the flux density around the TMS coil is incompletely sampled. Also, by controlling the number of magnetic dipoles used for constructing the theoretical coil model, it's shown that the method also enables a flexible trade-off between the accuracy and computational efficiency of the coil model when used in the numerical field calculations [2]. Based on the paper by Madsen et al. (2021) [2], the methodology for calculating the dipole moment and constructing the dipole coil is outlined as follows.

$$\mathbf{B}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r}) = \nabla \times \frac{\mu_0}{4\pi} \sum_i \frac{\mathbf{m}_i \times (\mathbf{r} - \mathbf{r}_i)}{|\mathbf{r} - \mathbf{r}_i|^3} = \frac{\mu_0}{4\pi} \sum_i \frac{1}{|\mathbf{r} - \mathbf{r}_i|^5} \left[3(\mathbf{r} - \mathbf{r}_i)(\mathbf{m}_i \cdot (\mathbf{r} - \mathbf{r}_i)) - \mathbf{m}_i |\mathbf{r} - \mathbf{r}_i|^2\right] \tag{1}$$

2.2.1 Model

To calculate dipole moments, it's considered the magnetic flux density described in equation (1), where:

- $\mathbf{B}(\mathbf{r})$ is the magnetic flux density $\mathbf{B}$ at point $\mathbf{r}$,
- $\nabla \times \mathbf{A}(\mathbf{r})$ is the curl of the magnetic vector potential $\mathbf{A}$ at point $\mathbf{r}$,
- μ_0 is the permeability of free space,
- $\mathbf{r}$ is the position where the magnetic flux density is calculated,
- $\mathbf{r}_i$ is the position vector of the i-th dipole,
- $\mathbf{m}_i$ is the magnetic dipole moment of the i-th dipole.

By splitting coordinates and dipole moments into Cartesian coordinate parts, the problem can be written as a linear superposition of dipole fields collected in the lead filed matrix $\mathbf{L_B}$, shown in equation (2),

$$\mathbf{b} = \mathbf{L_B} \cdot \mathbf{m} \tag{2}$$

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{bmatrix} \quad \mathbf{b} \in R \tag{3}$$

$$\mathbf{L_B} = \begin{bmatrix} L_{11} & L_{12} & \dots & L_{1n} \\ L_{21} & L_{22} & \dots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{m1} & L_{m2} & \dots & L_{mn} \end{bmatrix} \quad \mathbf{L_B} \in R \tag{4}$$

$$\mathbf{m} = \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \end{bmatrix} \quad \mathbf{m} \in R \tag{5}$$

where:

- $\mathbf{b}$ is the vector of flux density measurements,
- $\mathbf{L_B}$ is the lead field matrix,
- $\mathbf{m}$ is the dipole moment vector.

Therefore, the matrix product $\mathbf{L_B} \cdot \mathbf{m}$ gives the resulting flux density measurements $\mathbf{b}$ as a linear combination of the contributions from each dipole moment component.
To solve this problem the simple minimum norm estimates based on regularization with a prior shrinkage, which amounts to seeking the solutions to the problem that minimizes the convex cost function $C(\mathbf{m})$, expressed in equation (6).

$$C(\mathbf{m}) = \|\mathbf{b} - \mathbf{L}\mathbf{m}\|_{\mathbf{F}}^2 + \alpha |\mathbf{\Gamma} \mathbf{m}|^2 \tag{6}$$

Such implementation is chosen due to the ill-conditioning of the problem.

2.2.2 Sparse model

With the same method, it is also solved the linear system

$$\mathbf{a} = \mathbf{L_A} \cdot \mathbf{m} \tag{7}$$

where:

- $\mathbf{a}$ is the vectorized magnetic vector potential,
- $\mathbf{L_A}$ is the lead filed matrix.

This is used, with equation (2), to create the sparse dipoles coil with a lower number of dipoles and the same performance.
To obtain approximately equal contributions of the $\mathbf{A}$ and $\mathbf{B}$ fields during the sparsification of the dipole model, the simulated data and lead fields were normalized. This led to the linear problem:

$$\begin{bmatrix} \frac{\mathbf{a}}{\|\mathbf{a}\|} \\ \frac{\mathbf{b}}{\|\mathbf{b}\|} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{L}_A}{\|\mathbf{a}\|} \\ \frac{\mathbf{L}_B}{\|\mathbf{b}\|} \end{bmatrix} \mathbf{m}_{\text{sparse}}, \tag{8}$$

where:

- $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are the corresponding L_2 norms of these vectors:
  - $\|\mathbf{a}\| = \sqrt{\mathbf{a}^\top \mathbf{a}}$
  - $\|\mathbf{b}\| = \sqrt{\mathbf{b}^\top \mathbf{b}}$
- $\mathbf{m}_{\text{sparse}}$ represents the dipole moments of the sparse dipole expansion.

This system of equations allows for the efficient computation of $\mathbf{m}_{\text{sparse}}$, facilitating the reduction of the dipole model while maintaining accuracy in the representation of both the $\mathbf{A}$ and $\mathbf{B}$ fields.

2.3. MadVenture example

SimNibs released an example of their dipole coil [3] created on MagVenture MC-B70 coil in three dimensions [4].
They start with an initial dipole position distributed on a regular grid with 5 mm spacing, covering positions inside the coil casing (Figure 1). Initially, all dipole moments are set to unity, then, after resolving the problem (2), the real dipole moments are calculated (Figure 2).
Then, they created the sparse dipole coil (Figure 3) with the relative dipole moments (Figure 4).

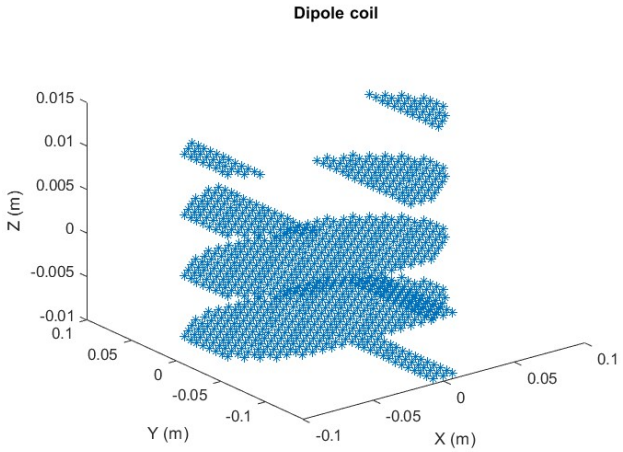

Figure 1: MagVenture MC-B70 coil: position of dipoles

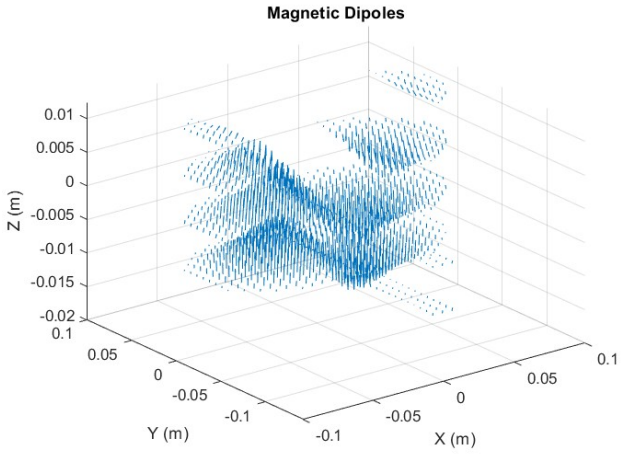

Figure 2: MagVenture MC-B70 coil: dipoles moments

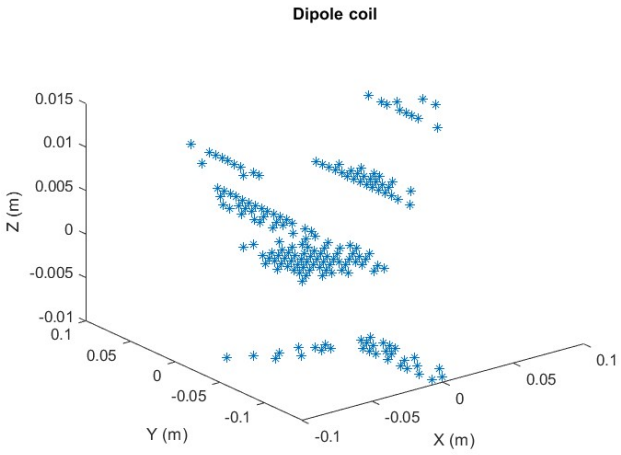

Figure 3: MagVenture MC-B70 sparse coil: points of dipoles

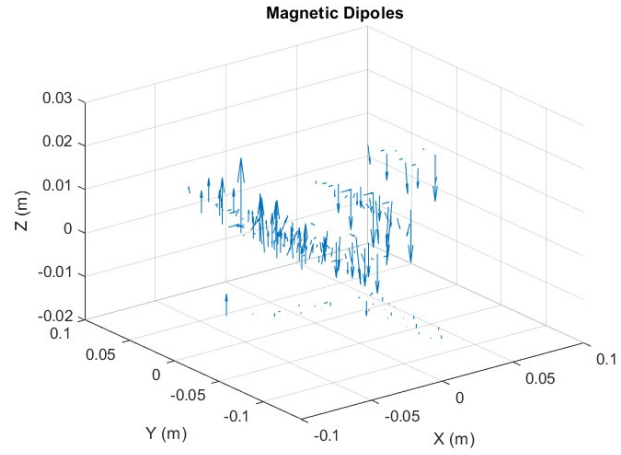

Figure 4: MagVenture MC-B70 sparse coil: dipoles moments

2.4. Advantages

The dipole modeling approach in SimNIBS offers several advantages [5]. It provides high precision in simulating magnetic fields, which is crucial for accurately predicting the spatial distribution and strength of induced electric fields. The flexibility of the dipole model allows for the simulation of various coil geometries and configurations used in TMS, supporting personalized treatment planning and experimental design. Moreover, dipole modeling is computationally efficient compared to more complex methods, facilitating rapid simulation and iterative testing of parameters.

3. Methods

The project aims to model an 8-figure coil using multiple magnetic dipoles and calculate the magnetic field using the Biot-Savart law for magnetic dipoles (1).
Subsequently, the calculated magnetic field is compared with the magnetic field obtained using the Biot-Savart law in its vector form from an 8-figure coil with line elements:

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int \frac{I\, d\mathbf{l} \times (\mathbf{r} - \mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|^3} \tag{9}$$

where:

- $\mathbf{B}(\mathbf{r})$ is the magnetic field $\mathbf{B}$ at point $\mathbf{r}$,
- μ_0 is the permeability of free space,
- I is the constant current,
- $d\mathbf{l}$ is the infinitesimal line element of the coil,
- $\mathbf{r}$ is the position where the magnetic field is calculated,
- $\mathbf{r}'$ is the position vector along the current element $d\mathbf{l}$, in this study, the midpoint of the line element is chosen.

The accuracy of the 8-figure coil approximation is evaluated using the mean square error, from equation (10), calculated on each magnetic field component.
The mean squared error (MSE) is given by:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (B_{seg_i} - B_{dip_i})^2 \tag{10}$$

where:

- N is the total number of elements,
- B_{seg_i} is the magnetic field calculated for line elements,
- B_{dip_i} is the magnetic field calculated with dipoles approximation.

Finally, the mean absolute error, referenced in equation (11), and the relative error, referenced in equation (12), is calculated on the magnitude of the magnetic field.
The mean absolute error (MAE) is calculated as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |B_{\text{Mseg}_i} - B_{\text{Mdip}_i}| \tag{11}$$

where:

- N is the total number of elements,
- B_{Mseg_i} is the magnitude of the magnetic field calculated with line elements 8-figure coil,
- B_{Mdip_i} is the magnitude of the magnetic field calculated with the dipole coil.

The relative error (relative E) is calculated as follows:

$$\text{relative E} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{B_{\text{Mseg}_i} - B_{\text{Mdip}_i}}{B_{\text{Mseg}_i}} \right| \times 100 \tag{12}$$

The mean absolute error provides a direct measure of the difference in magnitude between the magnetic fields calculated using the line elements of the 8-figure coil and the dipole coil, allowing for an assessment of the overall discrepancy in the magnetic field strengths. Instead, the relative error quantifies the accuracy of the dipole approximation compared with calculating the magnetic field using Biot-Savart law with linear elements.

3.1. 8-figure coil creation

Following [6], an 8-figure coil composed of two linked spirals is created for the simulations. The coil has an external radius of 0.05 meters, an internal radius of 0.032 meters, and a growth rate of 0.002 meters. The spirals are separated by a distance of 0.06 meters from the symmetrical center and are connected by a line from the inner lines of the two spirals. Additionally, two straight lines are completing the 8-figure coil (Figure 5). Using line elements, the entire coil is segmented into 1813 elements: 100 for each revolution of the spiral, 3 for the line between the two spirals and 5 for each straight line.

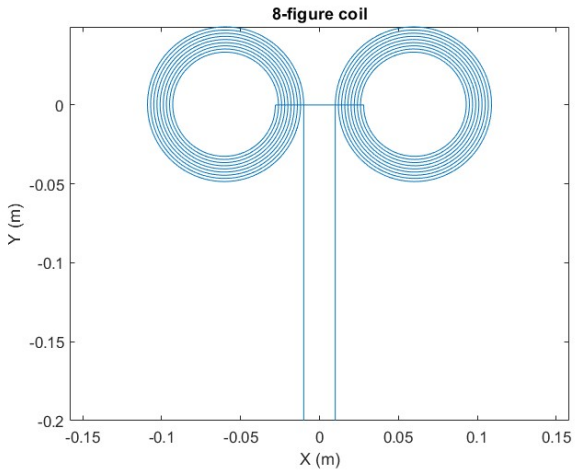

Figure 5: 8-figure coil

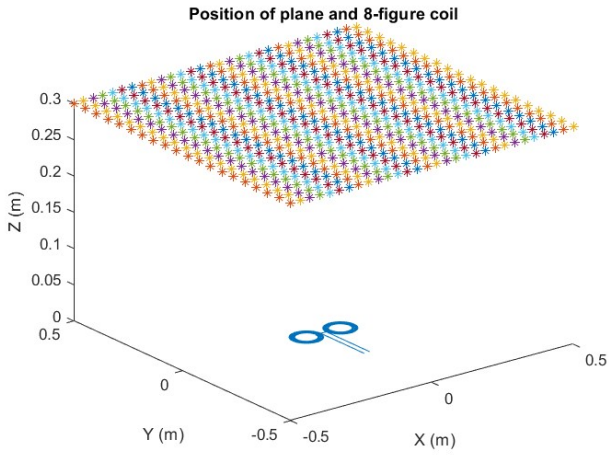

Figure 6: Position of the plane: 600 points

To model the 8-figure coil, three distinct vectors ($\mathbf{x_{coil}}$, $\mathbf{y_{coil}}$, $\mathbf{z_{coil}}$) represent the x, y, and z coordinates of the coil, each with a size of 1×1813. The vector of z coordinates is composed entirely of zeros because the 8-figure coil lies on the xy-plane.

$$\mathbf{x_{coil}} = \begin{bmatrix} -0.0100 & -0.0100 & \dots & 0.0100 \end{bmatrix}_{1\times 1813}$$

$$\mathbf{y_{coil}} = \begin{bmatrix} -0.2000 & -0.1500 & \dots & -0.2000 \end{bmatrix}_{1\times 1813}$$

$$\mathbf{z_{coil}} = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}_{1\times 1813}$$

3.2. Biot-Savart law with line elements

For the calculation of the Biot-Savart law at a single point, the law is implemented for a 2D figure and a point in three-dimensional space.
In MatLab [7], the three components of the magnetic field are stored in a single vector of size 1×3.
Furthermore, the magnetic field is calculated on a plane that is parallel to the coil and positioned 0.30 meters from the coil plane (Figure 6). This scenario allows for a clearer and more manageable analysis of the coil's magnetic field distribution, facilitating a detailed comparison between the approximations of line elements and dipoles. The plane consists of a two-dimensional rectangular grid of points with dimensions 1×1 meters. The points that compose the plane investigated are 300, 400, 450, 500, 600 and 750.

The Biot-Savart law (9) is implemented for the calculation of the magnetic field components at each point on the plane (Figure 7).

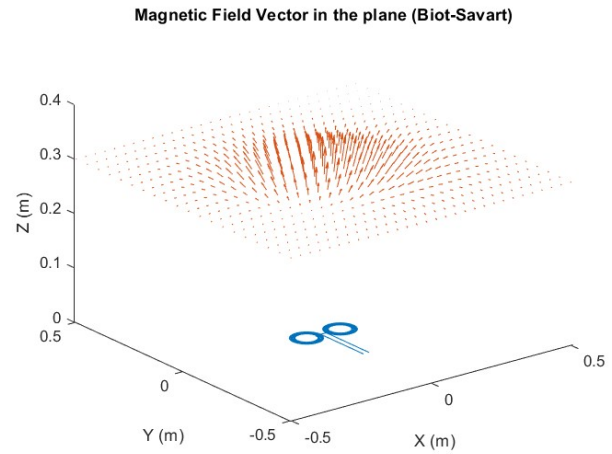

Figure 7: Magnetic field (T) calculated with 8-figure coil with line elements in a plane with 600 points

To create the plane positioned in xy-plane at $z = 0.30$, which represents the spatial distribution of points for magnetic field calculation, the plane is discretized into a grid with x and y coordinates ranging from -0.5 to 0.5 for both dimensions.
To model the plane, three distinct matrices ($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$) represent the coordinates of points on a plane in a three-dimensional space. The $\mathbf{Z}$ matrix is constant and set to 0.30. The general

matrix form for $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ is the following.

$$\mathbf{P} = \begin{bmatrix} \mathrm{p}_{11} & \mathrm{p}_{12} & \cdots & \mathrm{p}_{1n} \\ \mathrm{p}_{21} & \mathrm{p}_{22} & \cdots & \mathrm{p}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{p}_{m1} & \mathrm{p}_{m2} & \cdots & \mathrm{p}_{mn} \end{bmatrix} \quad \mathbf{P} \in R \qquad (13)$$

The total number of points in the plane, n_{plane}, is given by the product of these two quantities:

$$n_{\text{plane}} = m \times n.$$

Considering the 8-figure coil created with line elements and the plane defined, the magnetic field is calculated using the Biot-Savart law (9). The result is four vectors: three for each magnetic field component ($\mathbf{B_x}, \mathbf{B_y}, \mathbf{B_z}$) and one for the magnitude of the field $\mathbf{B}$.
The general matrix form for $\mathbf{B_x}, \mathbf{B_y}, \mathbf{B_z}, \mathbf{B}$ is the following.

$$\mathbf{F} = \begin{bmatrix} \mathrm{f}_{11} & \mathrm{f}_{12} & \dots & \mathrm{f}_{1n} \\ \mathrm{f}_{21} & \mathrm{f}_{22} & \dots & \mathrm{f}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{f}_{m1} & \mathrm{f}_{m2} & \dots & \mathrm{f}_{mn} \end{bmatrix} \quad \mathbf{F} \in R \qquad (14)$$

This calculation allows for a detailed mapping of the magnetic field generated by the 8-figure coil at specified locations within the plane.

3.3. Dipoles coil creation

A plane with dimensions 0.2×0.1 meters, which corresponds to the plane outlined by the two spirals of the 8-figure coil, is defined as a dipole plane in which dipoles follow different distributions (Figure 8).

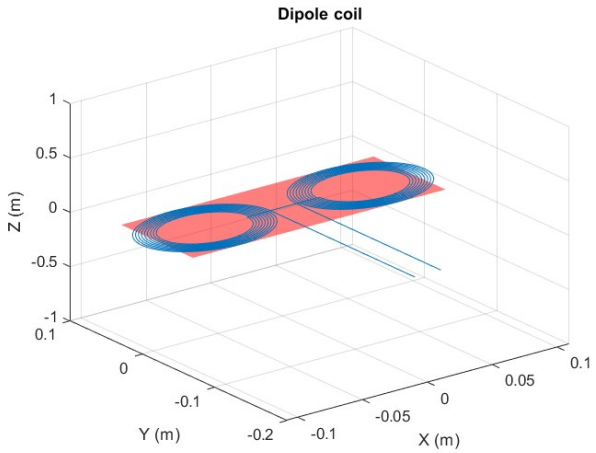

Figure 8: Dipoles surface overlapped with 8-figure coil with line elements

The points on the plane follow four types of distribution. For each distribution, the magnetic field is calculated to determine which one best approximates the magnetic field computed using the Biot-Savart law with line elements. The distributions are: Gauss-Legendre, as used in SimNIBS [8], Gauss-Chebyshev (Figure 9), and two distributions with equidistant points, one with the points farthest from each other and the other with the points closest. The Gauss-Legendre distribution is defined with function *lgwt* in MatLab [9] (Figure 10). The equidistant points are defined starting from the edge of the plane with a distance offset. Two distance offsets are considered: $dist = 0$ and $dist = 0.03$. With the first one, the points are located farthest from the center of the plane, in a plane 0.2×0.1 meters (Figure 11). With the second distance offset, the points are closest to the center of the plane and the defined plane is 0.14×0.07 meters (Figure 12).
The dipole coil configuration is generated using 100, 150, 200, 250, 375, 500, 750, 800, 1000, 1200 and 1600 points to compare the effectiveness of different distributions and their variations as the number of points changes.

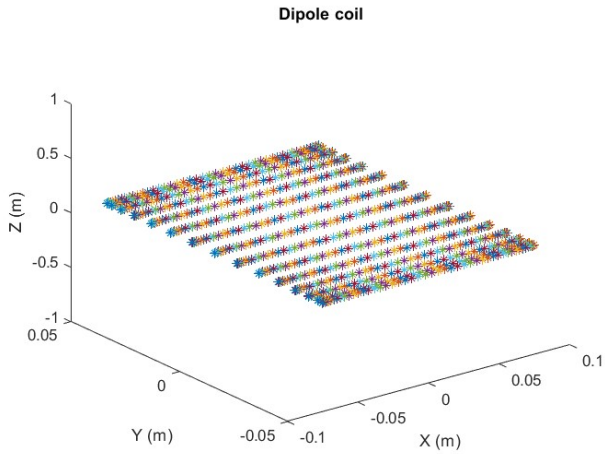

Figure 9: Dipole coil with 750 points positioned according to the Gauss-Chebyshev distribution

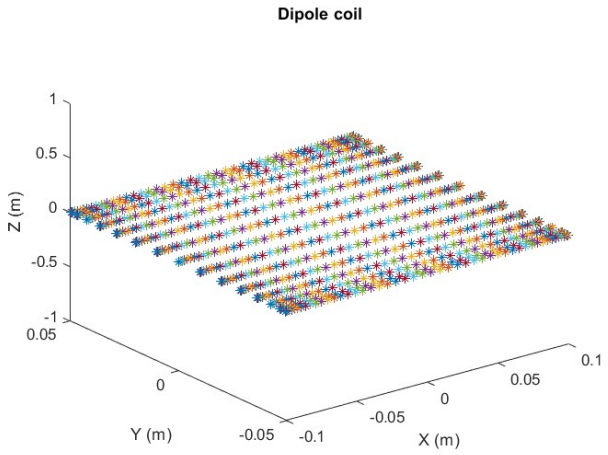

Figure 10: Dipole coil with 750 points positioned according to the Gauss-Legendre distribution

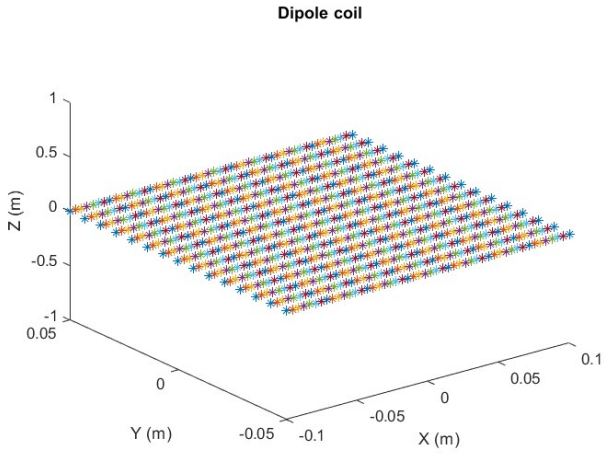

Figure 11: Dipole coil with 750 equidistant points farther from each other

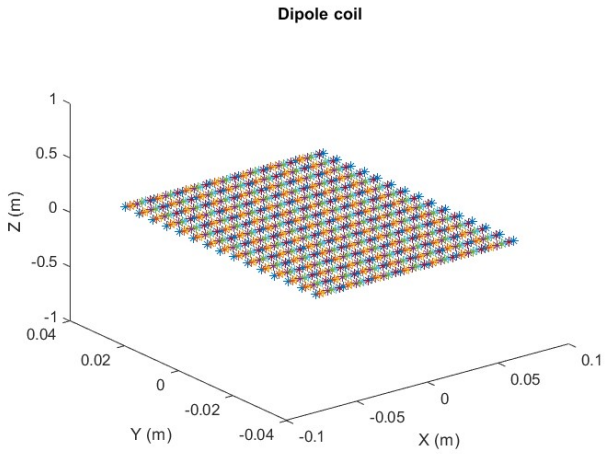

Figure 12: Dipole coil with 750 equidistant points closest from each other

To model the dipoles coil, we define three distinct matrices ($\mathbf{X_{dipoles}}$, $\mathbf{Y_{dipoles}}$, $\mathbf{Z_{dipoles}}$) representing the x, y, z coordinates of the coil. The general matrix form for $\mathbf{X_{dipoles}}$, $\mathbf{Y_{dipoles}}$, $\mathbf{Z_{dipoles}}$ is the following.

$$\mathbf{D} = \begin{bmatrix} \mathrm{d}_{11} & \mathrm{d}_{12} & \dots & \mathrm{d}_{1i} \\ \mathrm{d}_{21} & \mathrm{d}_{22} & \dots & \mathrm{d}_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{d}_{j1} & \mathrm{d}_{j2} & \dots & \mathrm{d}_{ji} \end{bmatrix} \quad \mathbf{D} \in R \tag{15}$$

The total number of dipoles, n_{dipoles}, is given by the product of these two quantities:

$$n_{\text{dipoles}} = j \times i.$$

For the dipole moments, a unitary value is assumed. So, a unitary vector of size 1×3 is created, representing all dipoles in the dipole coil.

3.4. Linear problem

Next, the following system of linear equations must be solved to calculate the actual dipole moments of the dipole coil

$$\mathbf{A} \cdot \mathbf{m} = \mathbf{b} \tag{16}$$

where:

- $\mathbf{A}$ is the coefficient matrix that relates the dipole moments to the magnetic field,
- $\mathbf{m}$ represents the vector of dipole moments,
- $\mathbf{b}$ is the resulting magnetic field.

This process aims to find the real dipole approximation that best represents the 8-figure coil created with line elements. The $\mathbf{A}$ matrix is computed by applying the Biot-Savart law with the dipoles coil (1) and normalizing the magnetic field by dividing by the unit dipole moments. The result is three distinct matrices ($\mathbf{A_x}, \mathbf{A_y}, \mathbf{A_z}$) one for each component x, y, z, each has the following structure.

$$\mathbf{A} = \begin{bmatrix} \mathrm{a}_{11} & \dots & \mathrm{a}_{1n_{dipoles}} \\ \mathrm{a}_{21} & \dots & \mathrm{a}_{2n_{dipoles}} \\ \vdots & \ddots & \vdots \\ \mathrm{a}_{n_{plane}1} & \dots & \mathrm{a}_{n_{plane}n_{dipoles}} \end{bmatrix} \quad \mathbf{A} \in R \tag{17}$$

The linear equation system is solved using the magnetic field calculated with Biot-Savart law

with line elements (9) and the computed matrix $\mathbf{A}$. The least squares method

$$\mathbf{m} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{18}$$

and the minimum norm least squares method, used in [2],

$$\mathbf{m} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \tag{19}$$

are employed to calculate dipole moments.
The first method is particularly effective when $\mathbf{A}$ is overdetermined ($n_{\text{plane}} \geq n_{\text{dipoles}}$), whereas the second method is suitable for underdetermined matrices or those of low rank.
The matrices $\mathbf{A}$ in the minimum norm least squares method are regularized by a factor of 1×10^{-4} ensuring that the modified matrices $\mathbf{A}$ are better conditioned for solving the equations.
For each magnetic field component $(\mathbf{B_x}, \mathbf{B_y}, \mathbf{B_z})$ computed using the Biot-Savart law with line elements (9) and its corresponding $\mathbf{A}$ matrix component $(\mathbf{A_x}, \mathbf{A_y}, \mathbf{A_z})$, the appropriate method is applied.
To fit the dimensions for solving the problem, the magnetic field is transposed into a vector of size $n_{\text{plane}} \times 3$ considering the position's index of the plane used for the calculation of the matrix $\mathbf{A}$. So $\mathbf{B_x}, \mathbf{B_y}, \mathbf{B_z}$ are now vectors, instead of matrices (14) and their structure is the following.

$$\mathbf{f} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n_{plane}-1} \end{bmatrix} \quad \mathbf{f} \in R \tag{20}$$

Thus, for each component, the $\mathbf{A}$ matrix is composed of n_{plane} rows and n_{dipoles} columns. The magnetic field vector $\mathbf{b}$ has dimensions $n_{\text{plane}} \times 1$, and the resulting dipole moment vector has dimensions $n_{\text{dipoles}} \times 1$.
The dipole moments are then calculated separately for each direction (x, y, z) and unified into a single matrix of size $n_{\text{dipoles}} \times 3$.
This process is repeated for each dipole distribution and the number of dipoles in the dipoles coil.

3.5. Biot-Savart law with dipoles

Considering real dipole moments and the plane defined, the magnetic field is calculated using the Biot-Savart law (Figure 13) with dipoles coil (1).

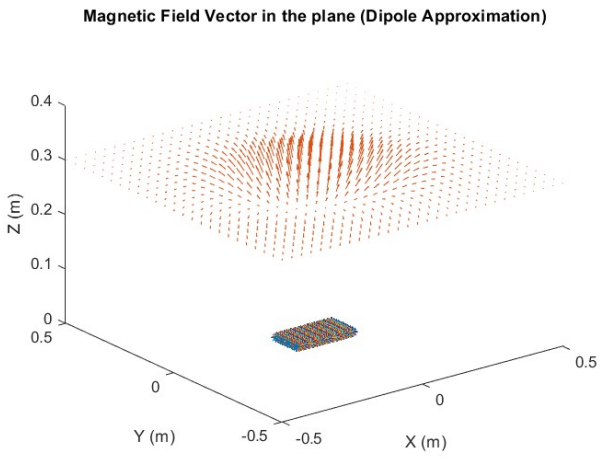

Figure 13: Magnetic field (T) calculated with dipoles farther from each other in a plane with 750 points

The result is four vectors: three for each magnetic field component $(\mathbf{B_x}, \mathbf{B_y}, \mathbf{B_z})$ and one for the magnitude of the field $\mathbf{B}$. The structure of each of these matrices is equal to matrix (14).
This process is repeated for each dipole distribution and the number of dipoles in the dipoles coil.

3.6. Evaluation

To evaluate the magnetic field calculated using Biot-Savart law with dipoles, the mean squared error (10) for all magnetic field components of the line elements coil and the different dipole coils is calculated. Also, the relative and mean absolute error (12-11) are evaluated over the magnitude of the magnetic field of the line elements coil and the different dipoles coil.
During the study, the combination of the number of dipoles and the number of points in the plane where the magnetic field $\mathbf{B}$ is calculated is investigated.
For least squares method, the following combinations are investigated:

- Plane with 300 points and dipole coil with 100, 150, 200 or 250 points,
- Plane with 400 points and dipole coil with 100, 150, 200, 250 or 375 points,
- Plane with 450 points and dipole coil with 100, 150, 200, 250 or 375 points,
- Plane with 500 points and dipole coil with 100, 150, 200, 250, 375 or 500 points,
- Plane with 600 points and dipole coil with 100, 150, 200, 250, 375 or 500 points,

- Plane with 750 points and dipole coil with 100, 150, 200, 250, 375, 500 or 750 points.

For minimum norm least squares method, the following combinations are investigated:

- Plane with 300 points and dipole coil with 375, 500, 750, 800, 1000, 1200 or 1600 points,
- Plane with 400 points and dipole coil with 500, 750, 800, 1000, 1200 or 1600 points,
- Plane with 450 points and dipole coil with 500, 750, 800, 1000, 1200 or 1600 points,
- Plane with 500 points and dipole coil with 750, 800, 1000, 1200 or 1600 points,
- Plane with 600 points and dipole coil with 750, 800, 1000, 1200 or 1600 points,
- Plane with 750 points and dipole coil with 800, 1000, 1200 or 1600 points.

4. Results

The magnetic field **B** and its components are computed for four differents dipole distributions: Gauss-Chebyshev, Gauss-Legendre, and two distributions with equidistant points. Among the equidistant point distributions, one has points positioned farther apart, while the other has points closer together.

For the given study, a series of dipole coils are created to different distributions, each containing 100, 150, 200, 250, 375, 500, 750, 800, 1000, 1200, and 1600 points. The dipole coils are generated using a grid with specific x and y dipole values: [10, 25, 50, 80] and [10, 15, 20] respectively. These coils are then evaluated on a plane with different points (300, 400, 450, 500, 600, and 750 points) created using specific values in the grid: 20 and 30 in the x-direction, and 15, 20, and 25 in the y-direction.

The relative error (relative E) and the mean absolute error (MAE) are compared between each distribution across all possible combinations of points on the plane and dipoles. Additionally, the magnetic field component that produced the highest error is investigated by examining each component's mean squared error (MSE).

Examining all the results collectively, it is immediately evident that the overdetermined systems exhibit lower error rates in terms of mean absolute error (Figures 14-17) and relative error (Figures 18-21) compared to the underdetermined systems. Notably, the error is significantly higher in the diagonal case involving the first underdetermined system.

From this point forward, in the graphs, "eq0" represents equidistant points that are farthest from each other, "eq003" denotes equidistant points that are closest to each other, "GL" stands for the Gauss-Legendre distribution, and "GC" refers to the Gauss-Chebyshev distribution.

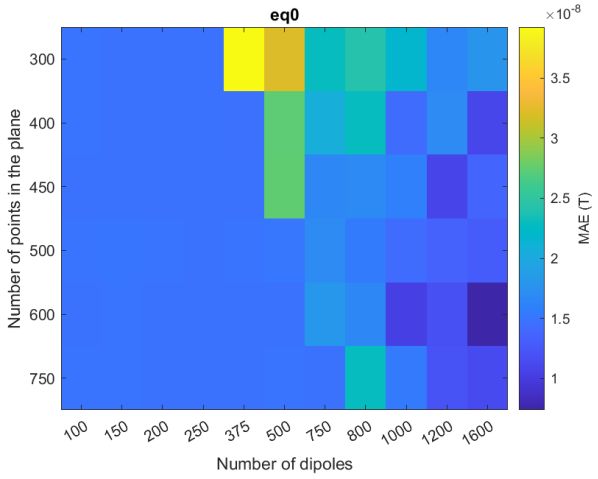

Figure 14: Mean absolute error for equidistant point farthest from each other

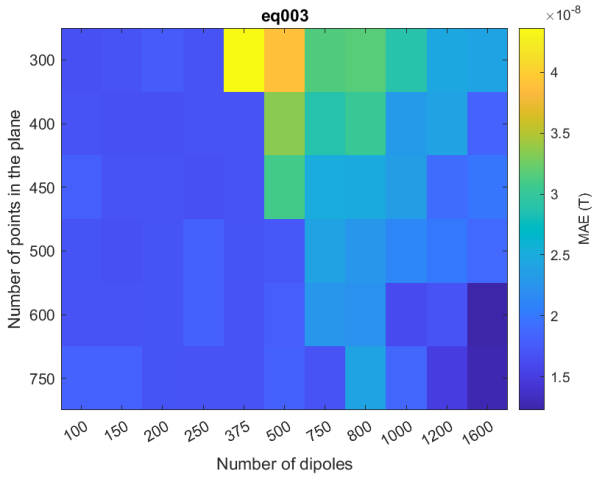

Figure 15: Mean absolute error for equidistant point closest from each other

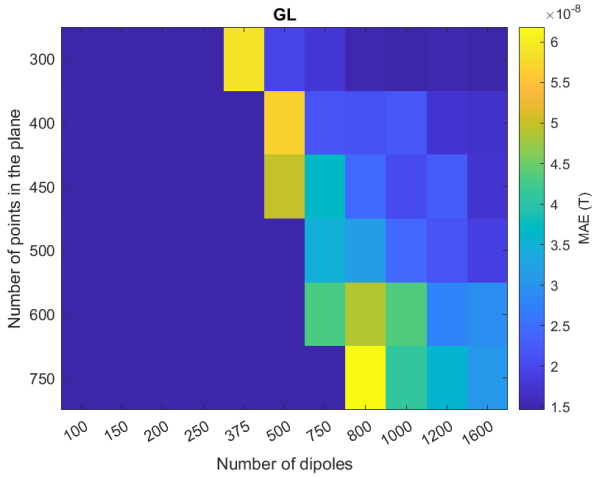

Figure 16: Mean absolute error for Gauss-Legendre distribution

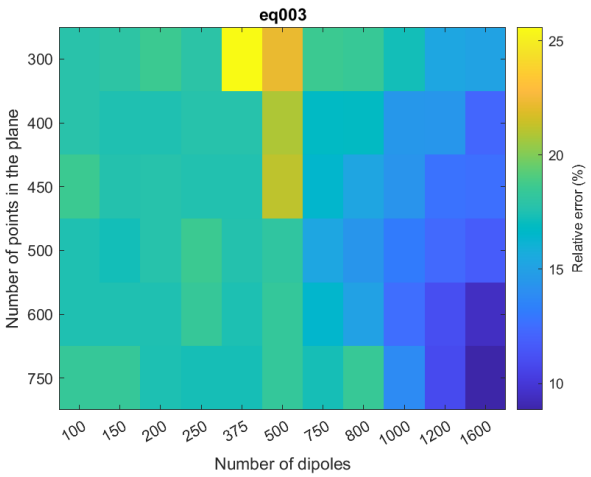

Figure 19: Relative error for equidistant point closest from each other

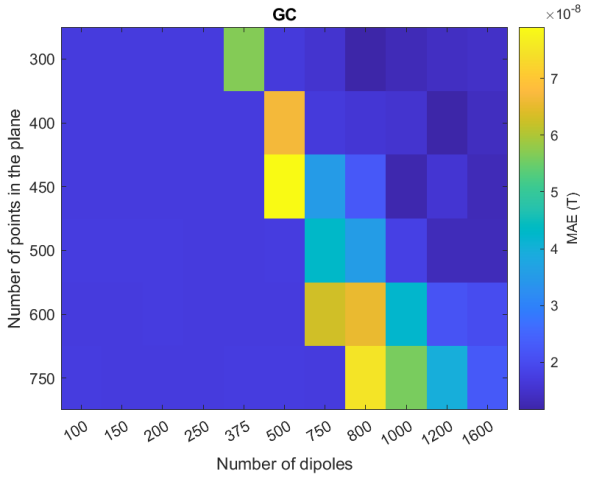

Figure 17: Mean absolute error for Gauss-Chebyshev distribution

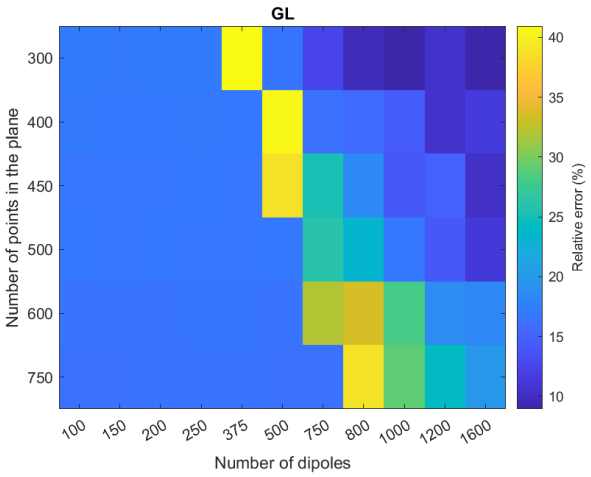

Figure 20: Relative error for Gauss-Legendre distribution

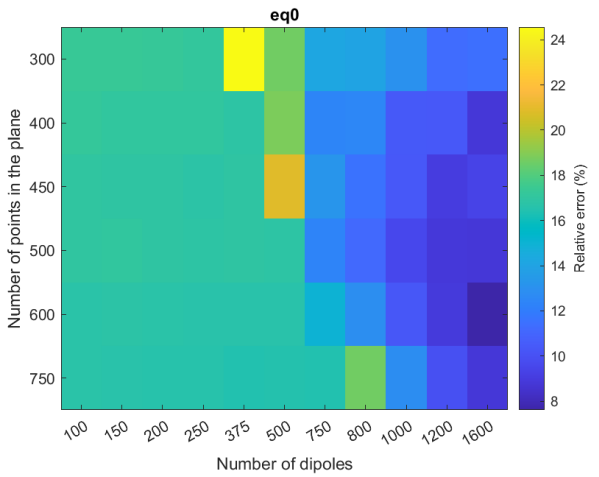

Figure 18: Relative error for equidistant point farthest from each other

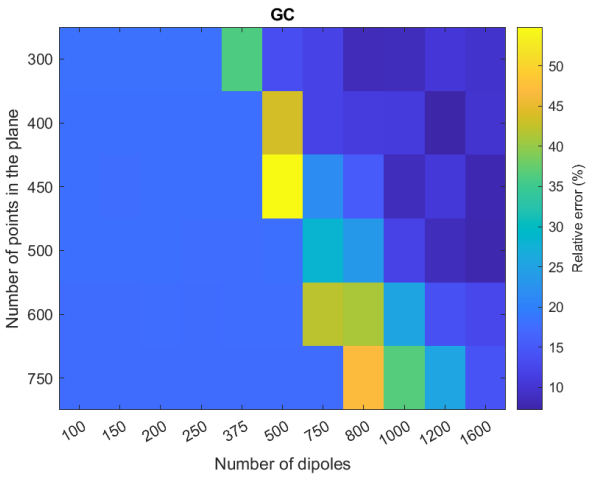

Figure 21: Relative error for Gauss-Chebyshev distribution

4.1. Least squares method

In this section, we focus on the least squares method to compare the distributions against each other, to assess whether some distributions exhibit greater error than others.

4.1.1 Plane with 300 points

For a plane consisting of 300 points, the absolute and relative errors (Figure 22-23) are calculated on the dipole coil using configurations of 100, 150, 200, and 250 points.

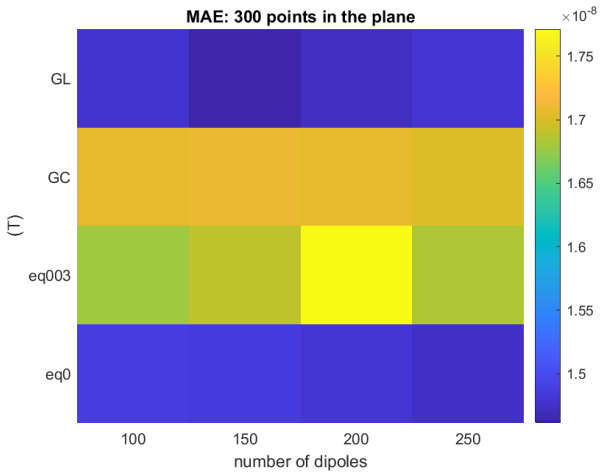

Figure 22: Mean absolute error for a plane with 300 points using least squares method

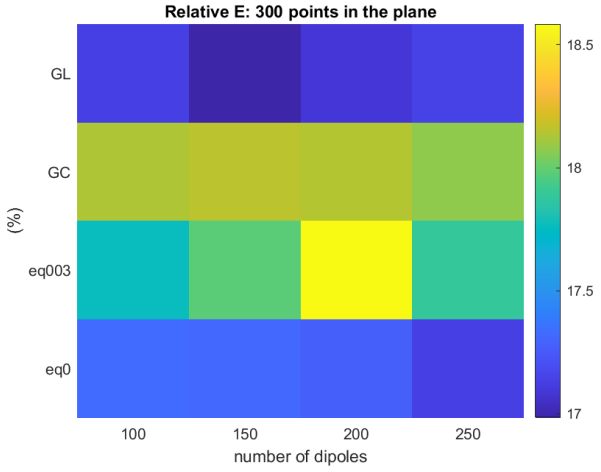

Figure 23: Relative error for a plane with 300 points using least squares method

For all dipole distributions, the relative error remains below 18.60%. Notably, for dipole coils with Gauss-Chebyshev, Gauss-Legendre and equidistant points farther from the center, the errors observed are consistently similar across varying numbers of dipoles in the coil.

4.1.2 Plane with 400 points

For a plane consisting of 400 points, the absolute and relative errors (Figure 24-25) are calculated on the dipole coil using configurations of 100, 150, 200, 250 and 375 points.

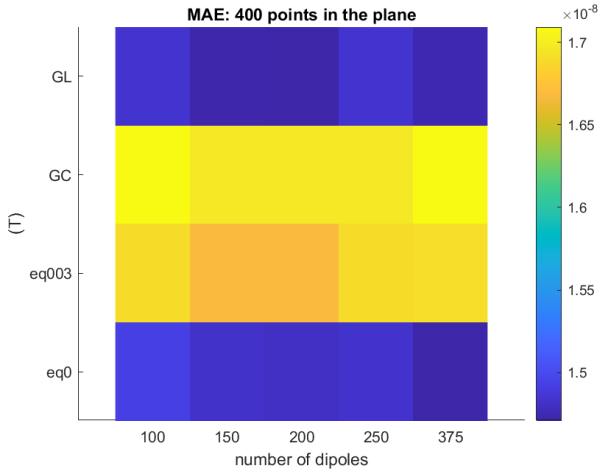

Figure 24: Mean absolute error for a plane with 400 points using least squares method

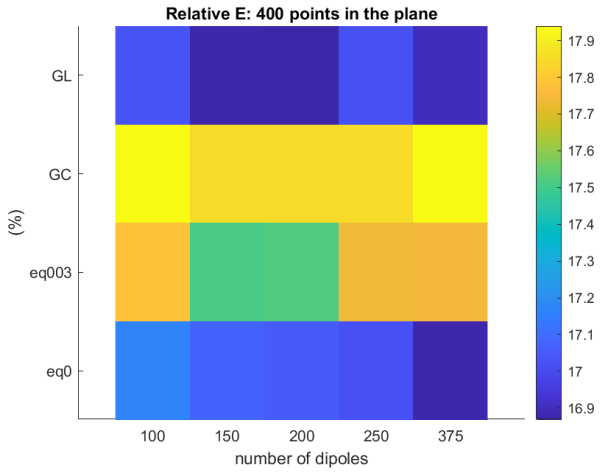

Figure 25: Relative error for a plane with 400 points using least squares method

For all dipole distributions, the relative error remains below 18%. Notably, for all dipole coils, the errors observed are consistently similar across varying numbers of dipoles in the coil.

4.1.3 Plane with 450 points

For a plane consisting of 450 points, the absolute and relative errors (Figure 26-27) are calculated on the dipole coil using configurations of 100, 150, 200, 250, and 375 points.

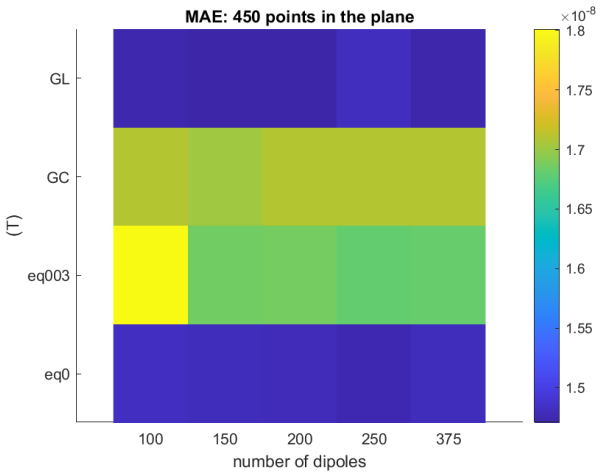

Figure 26: Mean absolute error for a plane with 450 points using least squares method

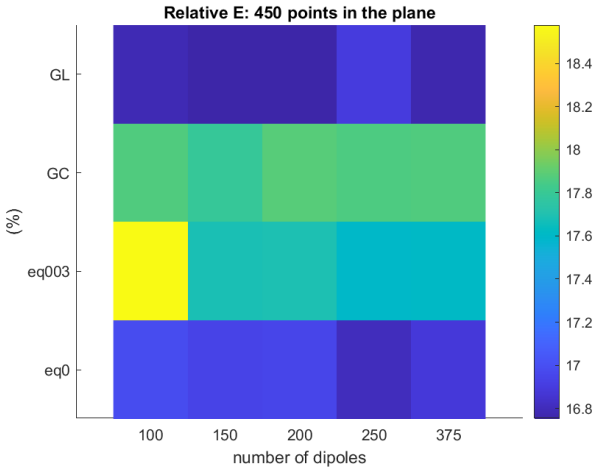

Figure 27: Relative error for a plane with 450 points using least squares method

For all dipole distributions, the relative error remains below 18.60%. Notably, for dipole coils with Gauss-Chebyshev, Gauss-Legendre and equidistant points farther from the center, the errors observed are consistently similar across varying numbers of dipoles in the coil.

4.1.4 Plane with 500 points

For a plane consisting of 500 points, the absolute and relative errors (Figure 28-29) are calculated on the dipole coil using configurations of 100, 150, 200, 250, and 375 points.

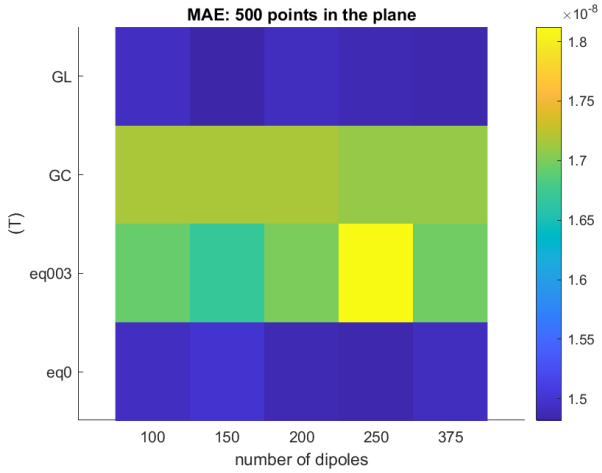

Figure 28: Mean absolute error for a plane with 500 points using least squares method

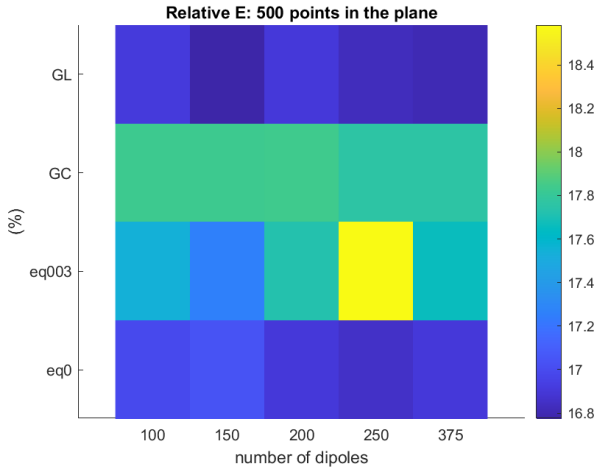

Figure 29: Relative error for a plane with 500 points using least squares method

For all dipole distributions, the relative error remains below 18.60%. Notably, for dipole coils with Gauss-Chebyshev, Gauss-Legendre and equidistant points farther from the center, the errors observed are consistently similar across varying numbers of dipoles in the coil.

4.1.5 Plane with 600 points

For a plane consisting of 600 points, the absolute and relative errors (Figure 30-31) are calculated on the dipole coil using configurations of 100, 150, 200, 250, 375, and 500 points. The dipole coils with 500 points are created with 50 points

on x-axis and 10 on y-axis and another with 25 on x-axis and 20 on y-axis.

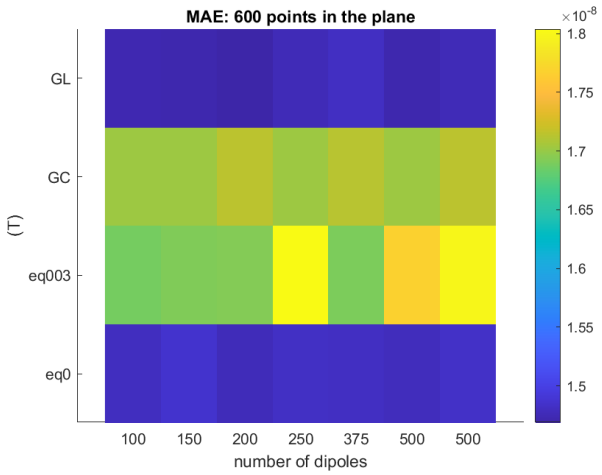

Figure 30: Mean absolute error for a plane with 600 points using least squares method

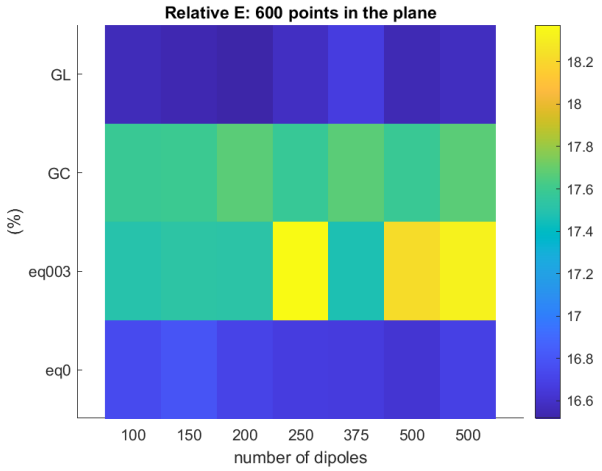

Figure 31: Relative error for a plane with 600 points using least squares method

For all dipole distributions, the relative error remains below 18.40%. Notably, for dipole coils with Gauss-Chebyshev, Gauss-Legendre and equidistant points farther from the center, the errors observed are consistently similar across varying numbers of dipoles in the coil.

4.1.6 Plane with 750 points

For a plane consisting of 750 points, the absolute and relative errors (Figure 32-33) are calculated on the dipole coil using configurations of 100, 150, 200, 250, 375, and 500 points.

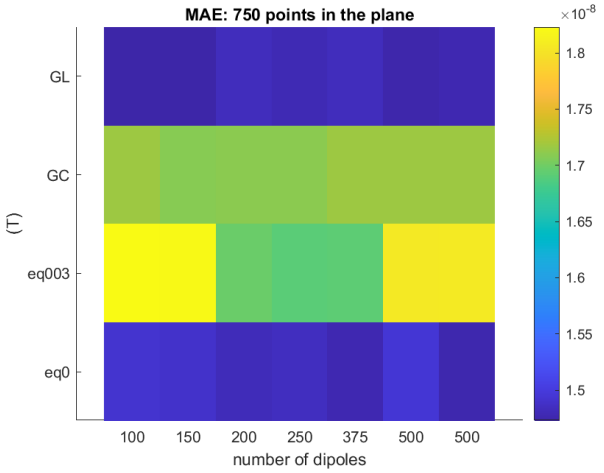

Figure 32: Mean absolute error for a plane with 750 points using least squares method

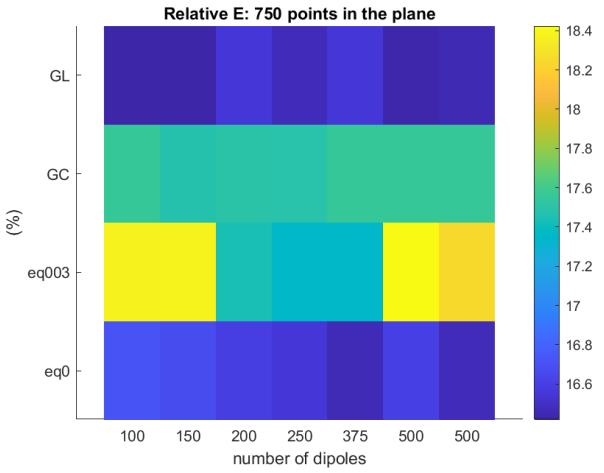

Figure 33: Relative error for a plane with 750 points using least squares method

For all dipole distributions, the relative error remains below 18.50%. Notably, for dipole coils with Gauss-Chebyshev, Gauss-Legendre and equidistant points farther from the center, the errors observed are consistently similar across varying numbers of dipoles in the coil.

In conclusion, the relative error is always between 18.60% and 14.45% and the mean absolute error is around 1.80×10^{-8}. In particular, the dipole coil with equidistant point closest to each other distribution has the highest relative error considering all distributions, the number of points in the plane and dipoles. The best dipoles coils are the Gauss-Legendre and equidistant points farthest from each other.

4.2. Minimum norm least squares method

In this section, we focus on the minimum norm least squares method to compare the distributions against each other, to assess whether some distributions exhibit greater error than others.

4.2.1 Plane with 300 points

For a plane consisting of 300 points, the absolute and relative errors (Figure 34-35) are calculated on the dipole coil using configurations of 375, 500, 750, 800, 1000, 1200, and 1600 points.

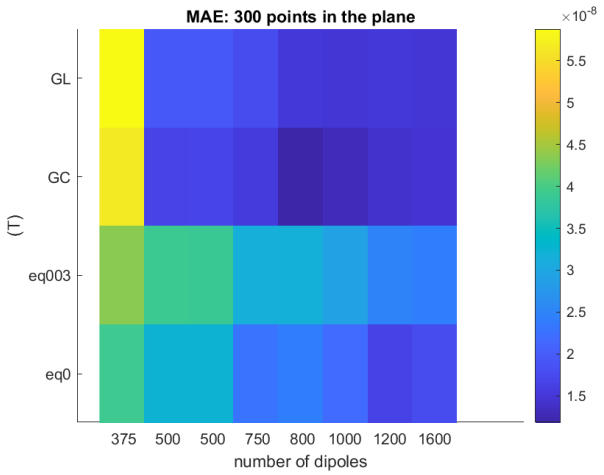

Figure 34: Mean absolute error for a plane with 300 points using minimum norm least squares method

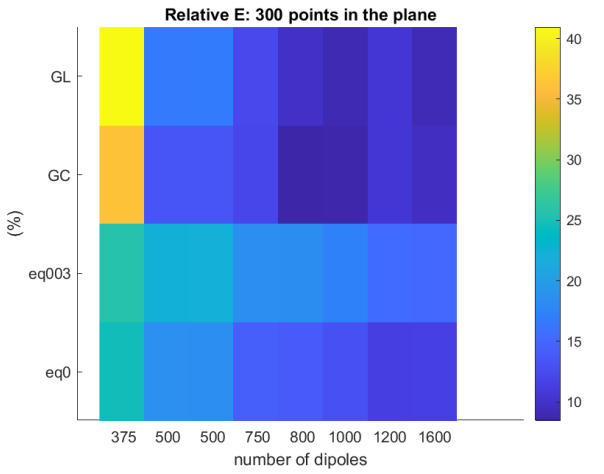

Figure 35: Relative error for a plane with 300 points using minimum norm least squares method

For all dipole distributions, the relative error remains below 45%. Notably, the errors decrease as the number of dipoles in the coils increases, leading to values of less than 25% at 500 dipoles and above.

4.2.2 Plane with 400 points

For a plane consisting of 400 points, the absolute and relative errors (Figure 36-37) are calculated on the dipole coil using configurations of 500, 750, 800, 1000, 1200, and 1600 points.

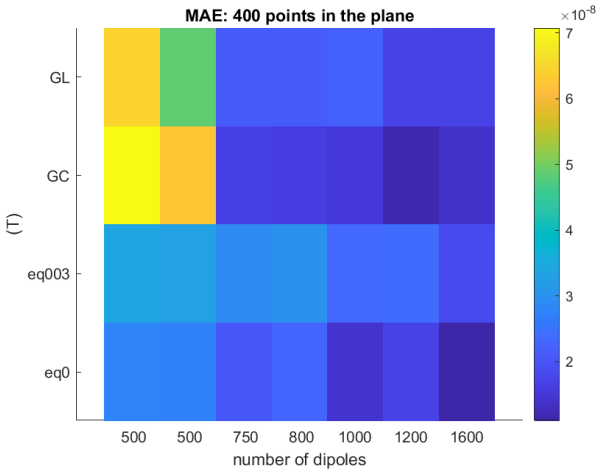

Figure 36: Mean absolute error for a plane with 400 points using minimum norm least squares method

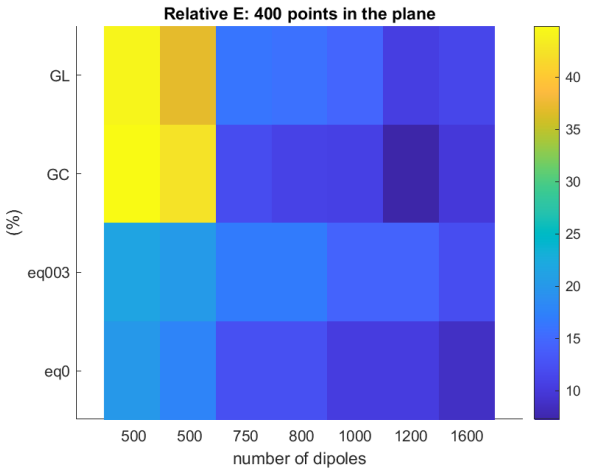

Figure 37: Relative error for a plane with 400 points using minimum norm least squares method

For all dipole distributions, the relative error remains below 45%. Notably, the errors decrease as the number of dipoles in the coils increases,

leading to values of less than 20% at 750 dipoles and above.

4.2.3 Plane with 450 points

For a plane consisting of 450 points, the absolute and relative errors (Figure 38-39) are calculated on the dipole coil using configurations of 500, 750, 800, 1000, 1200, and 1600 points.

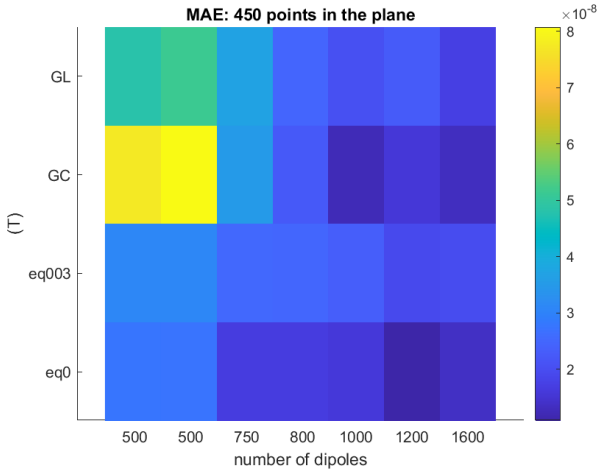

Figure 38: Mean absolute error for a plane with 450 points using minimum norm least squares method

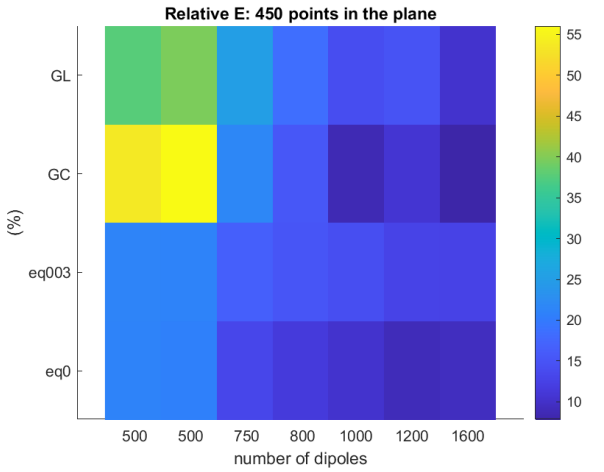

Figure 39: Relative error for a plane with 450 points using minimum norm least squares method

For all dipole distributions, the relative error remains below 60%. Notably, the errors decrease as the number of dipoles in the coils increases, leading to values of less than 25% at 750 dipoles and above.

4.2.4 Plane with 500 points

For a plane consisting of 500 points, the absolute and relative errors (Figure 40-41) are calculated on the dipole coil using configurations of 750, 800, 1000, 1200, and 1600 points.

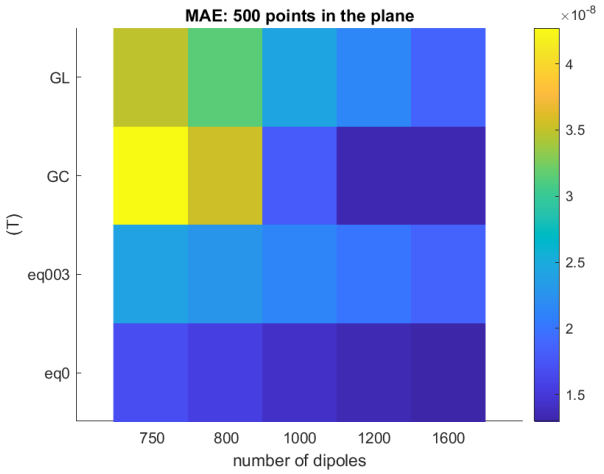

Figure 40: Mean absolute error for a plane with 500 points using minimum norm least squares method

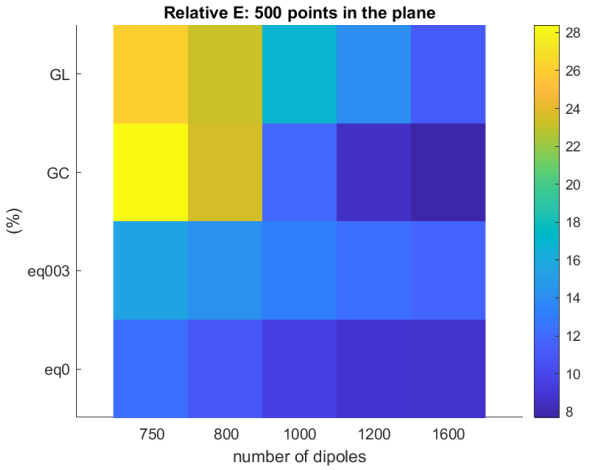

Figure 41: Relative error for a plane with 500 points using minimum norm least squares method

For all dipole distributions, the relative error remains below 30%. Notably, the errors decrease as the number of dipoles in the coils increases, leading to values of less than 20% at 1000 dipoles and above.

4.2.5 Plane with 600 points

For a plane consisting of 600 points, the absolute and relative errors (Figure 42-43) are calculated on the dipole coil using configurations of 750, 800, 1000, 1200, and 1600 points.

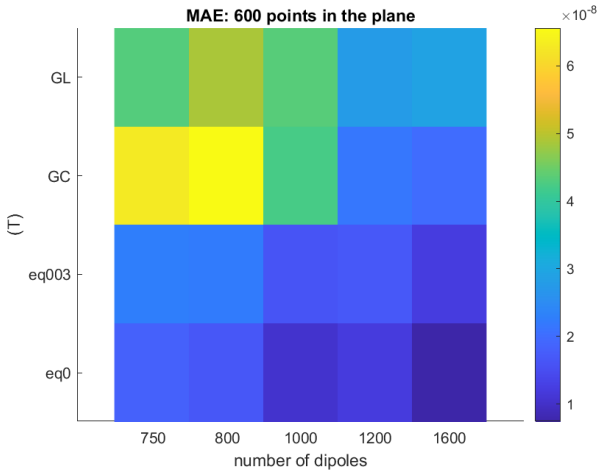

Figure 42: Mean absolute error for a plane with 600 points using minimum norm least squares method

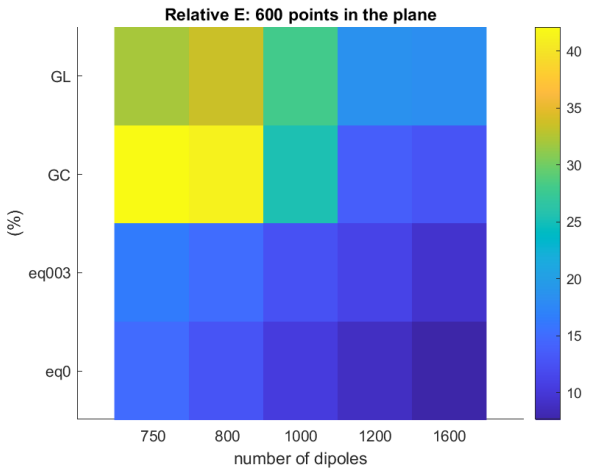

Figure 43: Relative error for a plane with 600 points using minimum norm least squares method

For all dipole distributions, the relative error remains below 45%. Notably, the errors decrease as the number of dipoles in the coils increases, leading to values of less than 30% at 1000 dipoles and above.

4.2.6 Plane with 750 points

For a plane consisting of 750 points, the absolute and relative errors (Figure 44-45) are calculated on the dipole coil using configurations of 800, 1000, 1200, and 1600 points.

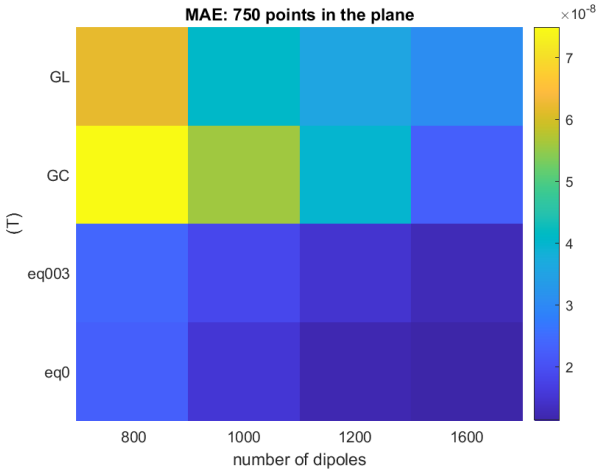

Figure 44: Mean absolute error for a plane with 750 points using minimum norm least squares method

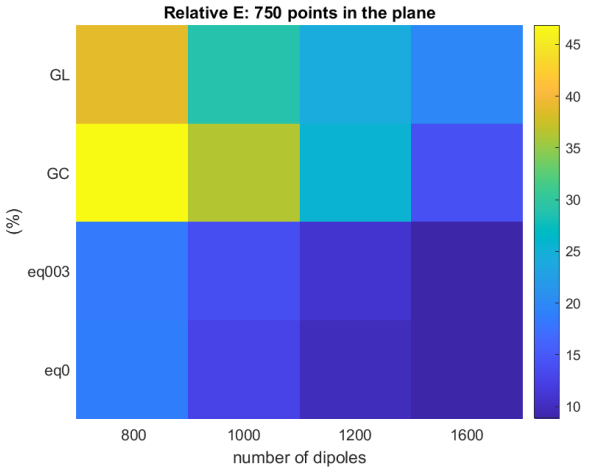

Figure 45: Relative error for a plane with 750 points using minimum norm least squares method

For all dipole distributions, the relative error remains below 50%. Notably, the errors decrease as the number of dipoles in the coils increases, leading to values of less than 25% at 1200 dipoles and above.

In conclusion, the relative error decreases as the number of dipoles in the coils increases: it is high as many dipoles as points in the plane, but

when the dipoles are higher the relative error is always less than 25%. The mean absolute error is around 8×10^{-8} with many dipoles similar to points in the plane, but it decreases around 2×10^{-8} as the number of dipoles increases. The best dipoles coils are equidistant points farthest and closest to each other where the relative error is always below 25%

Compared with the least squares method, there is greater variation in errors between different numbers of dipoles, which decreases as the number of dipoles increases. However, the trend of errors in the different distributions is quite similar as the number of points in the plane changes. The two methods are comparable considering the results obtained by the least squares method and those obtained with a large number of dipoles in the minimum norm least squares method where both relative error and mean absolute error are low.

4.3. Modificated 8-figure coil

Additionally, to limit the variation, a modified 8-figure coil is created, consisting solely of the two spirals and the connecting line (Figure 46). This modification aims to reduce the overall error in the magnetic field calculations.

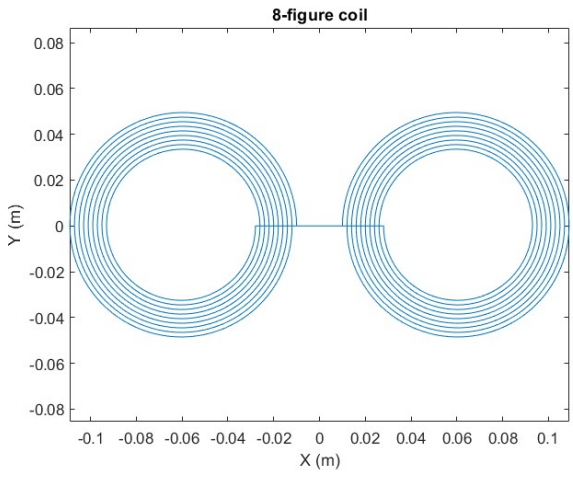

Figure 46: Modified 8-figure coil

Despite this effort, the relative error observed in the modified coil shows a consistent pattern, in fact, the error is equivalent when using the least squares method, but it is slightly lower compared to the original configuration using the minimum norm least squares method (Table 1).

8-figure coil	B with minimum norm least squares method (750 points in the plane, 1600 dipoles)	B with least squares method (500 points in the plane, 200 dipoles)
Complete	20.26%	16.90%
Modified	19.81%	16.91%

Table 1: Relative error of **B** field with Gauss-Legendre distribution for a dipole coil

This suggests that although the simplified coil design modifies the distribution of errors, it does not necessarily lead to an overall reduction in errors.

4.4. Mean squared error

Focusing on the mean squared error (MSE) of the individual magnetic field components, both in underdetermined and overdetermined systems, the z component has a higher MSE (Figure 47) for all dipoles distributions.

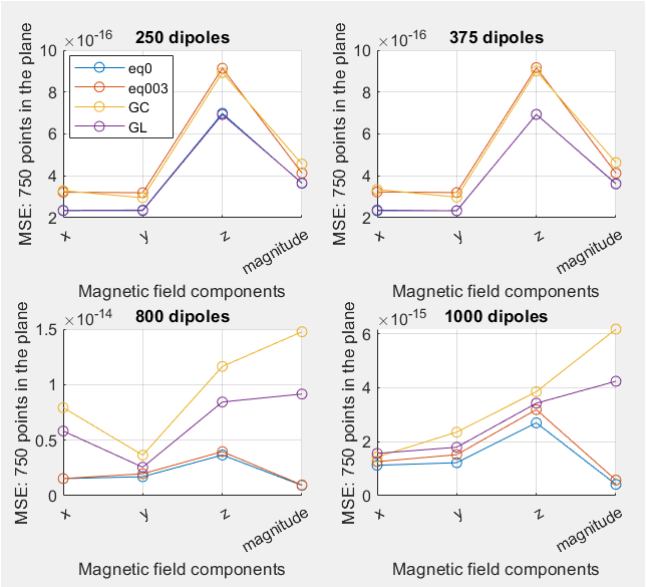

Figure 47: Mean squared error of magnetic field components and magnitude calculated on a plane with 750 points: for the upper graphs, overdetermined systems, and for the bottom graphs, underdetermined systems

4.5. Dipoles moments

Finally, examining the dipole moments, a similar magnetic vector distribution can be identified among the different dipole distributions used (Figure 48-51).

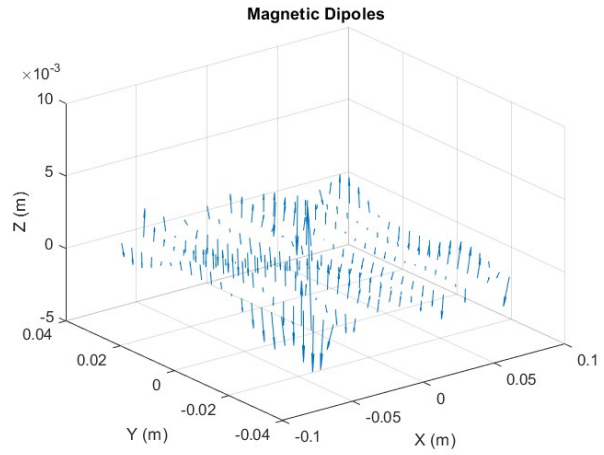

Figure 48: Dipoles moment of 200 equidistant points closest from each other

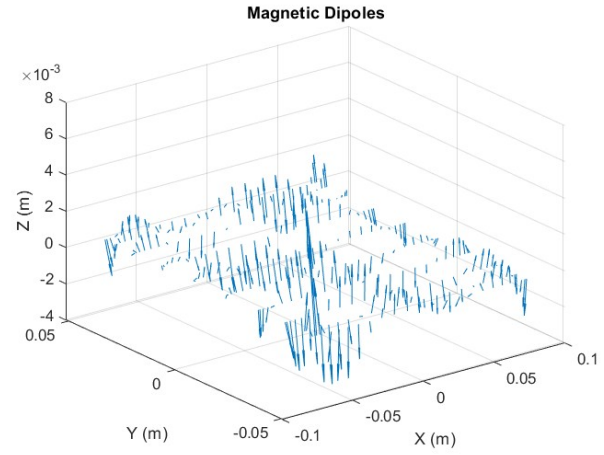

Figure 49: Dipoles moment of 250 points positioned according to the Gauss-Chebyshev distribution

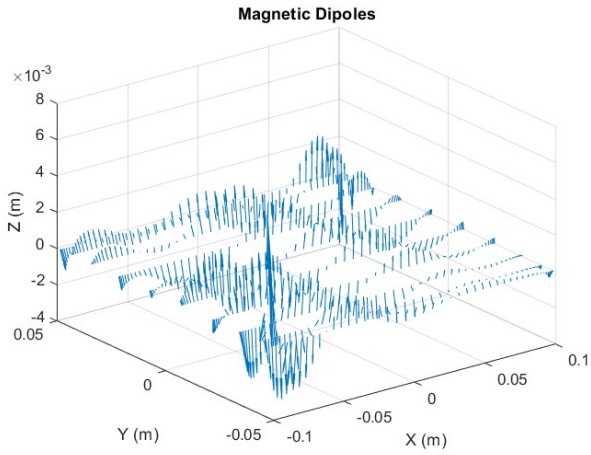

Figure 50: Dipoles moment of 500 dipoles positioned according to the Gauss-Legendre distribution

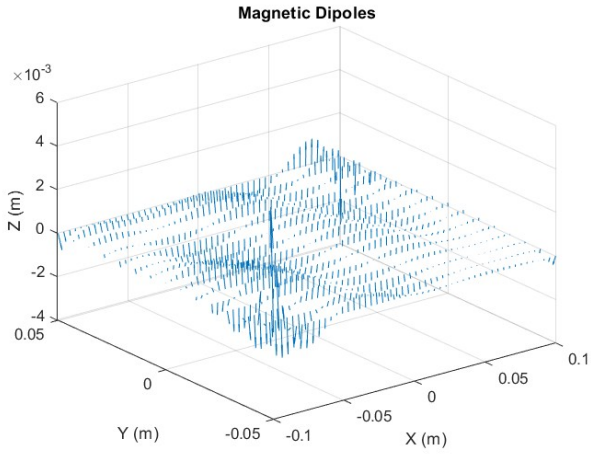

Figure 51: Dipoles moment of 750 equidistant points farther from each other

5. Conclusions

The overdetermined system shows more stable results as the number of dipoles varies. For the underdetermined systems, the value stabilizes more as the number of dipoles increases relative to the number of points on the plane.

In conclusion, the overdetermined system demonstrates stability as the number of dipoles varies, independent of the dipole distribution chosen. In contrast, underdetermined systems stabilize more with an increasing number of dipoles relative to the plane points and they are influenced by the dipole distribution chosen.

Using the least squares method the best dipoles coils are the Gauss-Legendre and equidistant

points farthest from each other, instead, for the minimum norm least squares method, the bests are equidistant points farthest and closest to each other, both methods in these cases reported relative error below 25% and an absolute one around 2×10^{-8}.
Finally, the dipole coil proves to be a reliable method given that the magnetic field **B** calculated with line elements and dipole coil are similar each other. This difference underscores the accuracy and effectiveness of the dipole coil.

6. Future developments

Future developments should aim to exploit advances in numerical methods, refine coil geometries, explore complex configurations, and improve user accessibility to advance the accuracy and applicability of magnetic field modeling for 8-digit coils.
One promising development involves incorporating more complex coil designs, such as investigating asymmetric coil geometries, variable coil shapes, or multi-coil configurations. Indeed, the main advantage of the dipole approximation lies in its ability to handle arbitrary coil geometries, including highly curved coils [5]. With this implementation, it is important to optimize the calculation of the magnetic field, potentially reducing computational time while maintaining accuracy. One effective approach is employing Fast Fourier Transform (FFT) techniques [10].

References

[1] Axel Thielscher, Andre Antunes, and Guilherme B Saturnino. "Field modeling for transcranial magnetic stimulation: A useful tool to understand the physiological effects of TMS?" In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2015, pp. 222–225. DOI: 10.1109/EMBC.2015.7318340.

[2] Kristoffer Hougaard Madsen, Maria Drakaki, and Axel Thielscher. "Electric field models of transcranial magnetic stimulation coils with arbitrary geometries: reconstruction from incomplete magnetic field measurements". In: *arXiv preprint arXiv:2112.14548* (2021).

[3] SimNibs. *dipole-fit*. Available at https://github.com/simnibs/dipole-fit. Accessed: June 17, 2024.

[4] *MagVenture MC-B70 Transcranial Magnetic Stimulation System*. Available at https://magventure.com/en_eur/products/mc-b70. Accessed: June 17, 2024.

[5] Maria Drakaki et al. "Database of 25 validated coil models for electric field simulations for TMS". In: *Brain stimulation* 15.3 (2022), pp. 697–706.

[6] Alessandra Paffi et al. "A computational model for real-time calculation of electric field due to transcranial magnetic stimulation in clinics". In: *International Journal of Antennas and Propagation* 2015.1 (2015), p. 976854.

[7] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: https://www.mathworks.com.

[8] Luis J Gomez, Moritz Dannhauer, and Angel V Peterchev. "Fast computational optimization of TMS coil placement for individualized electric field targeting". In: *Neuroimage* 228 (2021), p. 117696.

[9] *Greg von Winckel. Legendre-Gauss Quadrature Weights and Nodes*. Available at https://www.mathworks.com/matlabcentral/fileexchange/4540-legendre-gauss-quadrature-weights-and-nodes. Accessed: June 17, 2024.

[10] Hassan Yazdanian et al. "Fast evaluation of the Biot-Savart integral using FFT for electrical conductivity imaging". In: *Journal of Computational Physics* 411 (Mar. 2020), p. 109408. DOI: 10.1016/j.jcp.2020.109408.

A. MagVenture MC-B70 coil

These codes are related to the creation of dipole coil starting from MagVenture MC-B70 in section 2.3.

SimNibs code for creating MagVenture sparse dipole coil

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
 Copyright (C) 2021 Kristoffer H. Madsen

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>
"""

import numpy as np
import os
import coilfitter as cf
import sparsecoil as sc
import trimesh
print(os.getcwd())
## Load existing dipole model
ccdfile = 'coildata/MagVenture_MC-B70_dipole-fit.ccd'
dpos, dmoment = cf.readccd(ccdfile)

## Identify potential dipole positions

#load stl file with coil enclosure
coilmesh = trimesh.load_mesh('coildata/MagVenture_MC-B70_dipole-fit.stl')
#mesh in mm - scale to meters
coilmesh.apply_scale(0.001)
#determine coordinate bounds (min, max) for mesh coordinate
bounds = (np.min(coilmesh.vertices, 0), np.max(coilmesh.vertices, 0))
#set dipole density
dip_density = 7/1000.
#span out cartesian grid
x,y,z = [np.arange(bounds[0][i],bounds[1][i]+dip_density,dip_density)
        for i in range(3)]
xyz = np.array(np.meshgrid(*tuple([np.arange(bounds[0][i],bounds[1][i] +
                          dip_density,dip_density) for i in range(3)]),
                         indexing='ij'))
#use trimesh to figure out if positions are inside or outside the coilmesh
inmesh=coilmesh.contains(xyz.reshape(3,-1).T)
#reshape to dipole positions by 3 (x,y,z)
coilcoords=xyz.reshape(3,-1)[:,inmesh]

#get some points in a sphere in front of the coil and a plane behind
pfront = sc.get_sph_points(r=85,d=5)[:,:3000] / 1000.
pback = sc.pos_back(coilcoords, dz=0.015)
pos = np.concatenate((pfront, pback), axis=1).T
```

```
54
55 ## Get sparsified coil model
56 cpath = sc.sparsify(ccdfile, pos, coilcoords.T, max_n=500)
57
58 ## Produce dipoles vs. reconstruction tradeoff graph via CV
59 # First create some additional points in front of coil for CV
60 pos_cv = sc.get_sph_points(r=85, d=5)[:, :5000].T / 1000.
61 # Placehoder for coefficients of variation
62 R_cv = []
63 # Placehoder for number of active dipoles
64 n = []
65 # Calculate A field in CV positions
66 dip_pos0, dip_moment0 = cf.readccd(ccdfile)
67 Afull = sc.A_from_dipoles(dip_moment0, dip_pos0, pos_cv)
68
69 #loop over sparsification levels
70 for i in range(1,cpath.shape[1],1):
71     #find active dipoles
72     idx = np.any(cpath[:,i].reshape(3,-1)!=0,axis=0)
73     #calculate A from sparse model
74     Asparse = sc.A_from_dipoles(cpath[:,i].reshape(3,-1).T[idx], coilcoords.T[idx], pos_cv)
75     #coefficient of variation
76     R_cv.append(1-((Asparse-Afull)**2).sum()/(Afull**2).sum())
77     #active dipoles
78     n.append(idx.sum())
79
80 # The following section serves to calculate the error on the surface of spheres
81 # at a range of distances from the coil (to approximate a head shape). It then
82 # calculates the fractional error for each of these surfaces with increasing
83 # number of dipoles. It then stops when the required error tolerance has been
84 # reached for all distances (max over distances). This serves to control the
85 # error at the range of distances rather than overall error which would
86 # effectively only consider distances very close to the coil.
87 # This is in accordance with what is done in the paper.
88
89 dmin = 5 #minimum distance
90 dmax = 84. #maximum distance
91 dres = 0.25 #distance resolution
92 tol = 1e-2 #required tolerance
93 #Vector of distances
94 dists = np.arange(dmin, dmax + dres, dres)
95 # Generate 500 surface point for each distance
96 epos = np.concatenate([sc.get_surf_points(r=85, d=d, N_gen=1000)[:,:500,None] for d in
       dists], axis=2)
97 # Calculate A field for full model at all distances
98 Afull = sc.A_from_dipoles(dip_moment0, dip_pos0, epos.reshape(3, -1).T)
99 Afull = Afull.reshape(epos.shape[1:] + (3,))
100 # Place holder for errors
101 errA = []
102 # Loop over sparsification levels
103 for i in range(1,cpath.shape[1],1):
104     #find active dipoles
105     idx = np.any(cpath[:,i].reshape(3,-1)!=0,axis=0)
106     #calculate A from sparse model per distance
107     Asparse = sc.A_from_dipoles(cpath[:, i].reshape(3, -1).T[idx], coilcoords.T[idx],
          epos.reshape(3, -1).T)
108     Asparse = Asparse.reshape(epos.shape[1:] + (3,))
109     errA.append(((Asparse - Afull) ** 2).sum((0, 2)) / (Afull ** 2).sum((0, 2)))
```

```
    if np.max(errA[-1])<tol:
        break

# Produce plot of residual variance vs. number of active dipoles
import matplotlib.pyplot as plt
plt.figure()
plt.subplot(2,1,1)
plt.plot(n,1-np.array(R_cv))
plt.plot(n[i-1],1-R_cv[i-1],'ro')
plt.yscale('log')
plt.xlabel('# dipoles')
plt.ylabel(r'Fraction of residual variance')

plt.subplot(2,1,2)
plt.plot(n[:i],np.max(np.array(errA),1))
plt.plot(n[i-1],np.max(errA[i-1]),'ro')
plt.plot([n[0],n[i-1]],[tol,tol],'r--')
plt.yscale('log')
plt.xlabel('# dipoles')
plt.ylabel(r'Max. fraction of residual variance')

# Extract sparsified dipole model
dip_moment_sparse = cpath[:,i].reshape(3,-1).T[idx]
dip_pos_sparse = coilcoords.T[idx]
# Save sparse model
sparseccdfile = 'coildata/MagVenture_MC-B70_dipole-fit-sparse.ccd'
cf.writeccd(sparseccdfile, dip_pos_sparse, dip_moment_sparse)

print(f'Sparsified ccd file contains {dip_pos_sparse.shape[0]} active dipoles with residual
    error ~{100*(1-R_cv[i-1]):.3f} % compared to non-sparse model')
```

sparsify-MCB70-sph.py

Main code for visualizing the MagVenture dipoles coils

```
close all
clear
clc

%% Load ccd file

% SimNIBS creates a figure-eight coil with all points or in a sparse version.
% Set complete = 0 to visualize the sparse version.
complete = 0;

if complete
    % Load ccd file for complete coil
    opts = detectImportOptions('dipole_fit/coildata/MagVenture_MC-B70_dipole-fit.ccd',
        'FileType', 'text', 'Delimiter', ' ');

    % Specify the variable names
    opts.VariableNames = {'CenterX', 'CenterY', 'CenterZ', 'DirX', 'DirY', 'DirZ'};

    T = readtable('dipole_fit/coildata/MagVenture_MC-B70_dipole-fit.ccd', opts);
else
    % Load ccd file for sparse coil
    opts =
        detectImportOptions('dipole_fit/coildata/MagVenture_MC-B70_dipole-fit-sparse.ccd',
        'FileType', 'text', 'Delimiter', ' ');
```

```
22
23      % Specify the variable names
24      opts.VariableNames = {'CenterX', 'CenterY', 'CenterZ', 'DirX', 'DirY', 'DirZ'};
25
26      T = readtable('dipole_fit/coildata/MagVenture_MC-B70_dipole-fit-sparse.ccd', opts);
27  end
28
29  %% Visualization of dipole coil
30
31  figure
32  quiver3(T.CenterX, T.CenterY, T.CenterZ, T.DirX, T.DirY, T.DirZ);
33  xlabel('X (m)');
34  ylabel('Y (m)');
35  zlabel('Z (m)');
36  title('Magnetic Dipoles');
37
38  figure
39  plot3(T.CenterX, T.CenterY, T.CenterZ, '*');
40  xlabel('X (m)');
41  ylabel('Y (m)');
42  zlabel('Z (m)');
43  title('Dipole coil');
```

visualization-magVenture.m

B. Biot-Savart law calculation

These codes are related to Biot-Savart law calculation and the implementation of dipole coil related to section 3.

Function to create the 8-figure coil with line elements

```
1  function [x_coil, y_coil, z_coil, n_segments] = create_8_figure_coil(radius_ext, radius_in,
       c, center_dist, n_pts_l1, n_pts_l2, complete)
2
3  %{
4
5  %%% INPUT %%%
6  radius_ext: measure of external radius
7  radius_in: measure of internal radius:
8  c: incerement per revolution
9  center_dist: distance from the center
10 n_pts_l1: number of point in the line between the two spiral
11 n_pts_l2: number of point in the two straight lines
12
13 % Impose it to 1 if you want a complete 8-figure coil with the two straght
14 % lines
15 complete: logical value for create a complete 8-figure coil
16
17
18 %%% OUTPUT %%%
19 x_coil: OPTIONAL, X coordinate of 8-figure coil
20 y_coil: OPTIONAL, Y coordinate of 8-figure coil
21 z_coil: OPTIONAL, Z coordinate of 8-figure coil
22 n_segments: number of segments in 8-figure coil
23
24 %}
25
26 n_rev = (radius_ext - radius_in)./c; % number of revolutions
```

```
n_pts = n_rev*100; % number of points in a spiral fixed at 100 points
i = linspace(0, n_rev, n_pts);

% Spiral on the left
x_left = (radius_in + c*i).* cos(2*pi*i) - center_dist;
y_left = -(radius_in + c*i).* sin(2*pi*i);
x_left = flip(x_left);
y_left = flip(y_left);

% Spiral on the right
x_right = -(radius_in + c*i).* cos(2*pi*i) + center_dist;
y_right = -(radius_in + c*i).* sin(2*pi*i);

% Link line
k = 0.01;
x_l = linspace(-(center_dist-radius_in-k), center_dist-radius_in-k, n_pts_l1);
y_l = zeros([1, n_pts_l1]);

% Straight line
x_sll = -(center_dist-radius_ext)*ones([1, n_pts_l2]);
y_sll = linspace(0, -(radius_ext+0.15), n_pts_l2);
x_slr = (center_dist-radius_ext)*ones([1, n_pts_l2]);
y_slr = linspace(0, -(radius_ext+0.15), n_pts_l2);
y_sll = flip(y_sll);

% Create one figure
% You can choose if create a complete 8-figure coil with the two straight
% lines imposing complete = 1, or create a 8-figure coil with just the two
% spirale and the link line.
if complete
    x_coil = [x_sll, x_left, x_l, x_right, x_slr];
    y_coil = [y_sll, y_left, y_l, y_right, y_slr];
else
    x_coil = [x_left, x_l, x_right];
    y_coil = [y_left, y_l, y_right];
end
n_segments = size(x_coil,2);
z_coil = zeros(1,n_segments);

end
```

create-8-figure-coil.m

Function to calculate magnetic field on one point with Biot-Savart law with line elements

```
function [B_point] = BS_one_point(P, deltal, midpoints, I, u0, fig, x_coil, y_coil, z_coil)

%{

%%% INPUT %%%
P: point in which calculate magnetic field
deltal: differential length elements
midpoints: midpoints of each differential length element
I: constant current
u0: permeability
fig: logical value for figures visualization
% THE COORDINATES OF THE COIL ARE OPTIONAL IF FIGURE IS EQUAL TO 0
x_coil: OPTIONAL, X coordinate of 8-figure coil
y_coil: OPTIONAL, Y coordinate of 8-figure coil
```

```
z_coil: OPTIONAL, Z coordinate of 8-figure coil

%%% OUTPUT %%%
B_point: magnetic field component calculated in point P wiht line elements

%}

% Calculate vectors r and distances R
r = P - midpoints;
R = sqrt(sum(r .^ 2, 2));

% Define constant
cost = I*u0/(4*pi);

% Biot-Savart law
prodVect = cross(deltal, r, 2);
num = cost .* prodVect;
B_point = sum(num ./ R.^3, 1);

if fig
    % For visualization the magnetic field B is moltiplicated by a power of
    % 10 to increase its value
    B_point = B_point*10^4;

    figure(1)
    plot3(x_coil, y_coil, z_coil)
    hold on
    plot3(P(1), P(2), P(3), '*')
    title('Position of point P and 8-figure coil')

    figure(2)
    plot3(x_coil, y_coil, z_coil);
    hold on;
    quiver3(P(1), P(2), P(3), B_point(1), B_point(2), B_point(3), 'MaxHeadSize', 2,
        'AutoScale', 'off');
    xlabel('x');
    ylabel('y');
    zlabel('z');
    axis equal
    title('Magnetic Field Vector at point P');
end

end
```

BS-one-point.m

Function to calculate magnetic field on a plane with Biot-Savart law with line elements

```
function [BX_biot_savart, BY_biot_savart, BZ_biot_savart, B_magnitude] =
    BS_plane(x_point_plane, y_point_plane, ...
    X, Y, Z, deltal, midpoints, I, u0, fig, x_coil, y_coil, z_coil)

%{

%%% INPUT %%%
x_point_plane: Number of points for x side in the plane
y_point_plane: Number of points for y side in the plane
X: X coordinates of plane in with you want to calculate magnetic field
Y: Y coordinates of plane in with you want to calculate magnetic field
```

```
Z: Z coordinates of plane in with you want to calculate magnetic field
deltal: differential length elements
midpoints: midpoints of each differential length element
I: constant current
u0: permeability
fig: logical value for figures visualization
% THE COORDINATES OF THE COIL ARE OPTIONAL IF FIGURE IS EQUAL TO 0
x_coil: OPTIONAL, X coordinate of 8-figure coil
y_coil: OPTIONAL, Y coordinate of 8-figure coil
z_coil: OPTIONAL, Z coordinate of 8-figure coil

%%% OUTPUT %%%
Magnetic field and its componentes calculated in the plane [X, Y, Z]
BX_biot_savart: X component of magnetic field calculated with line elements
BY_biot_savart: Y component of magnetic field calculated with line elements
BZ_biot_savart: Z component of magnetic field calculated with line elements
B_biot_savart: magnetic field calculated with line elements

%}

% Initialization of magnetic field components
BX_biot_savart = zeros(y_point_plane, x_point_plane);
BY_biot_savart = zeros(y_point_plane, x_point_plane);
BZ_biot_savart = zeros(y_point_plane, x_point_plane);

% Define constant
cost = I*u0/(4*pi);

% Calculate Biot-Savart law
for k = 1:x_point_plane
    for j = 1:y_point_plane
        r_vecs = [X(j, k) - midpoints(:, 1), Y(j, k) - midpoints(:, 2), Z(j, k) -
            midpoints(:, 3)];
        R = sqrt(sum(r_vecs .^ 2, 2));
        valid_R = R > 0;
        r_vecs = r_vecs(valid_R, :);
        dL_valid = deltal(valid_R, :);
        R_valid = R(valid_R);
        dB = cost * cross(dL_valid, r_vecs, 2) ./ (R_valid.^3);
        B_field = sum(dB, 1);
        BX_biot_savart(j, k) = B_field(1);
        BY_biot_savart(j, k) = B_field(2);
        BZ_biot_savart(j, k) = B_field(3);
    end
end

% Combine magnetic field components into magnitude
B_magnitude = sqrt(BX_biot_savart.^2 + BY_biot_savart.^2 + BZ_biot_savart.^2);

if fig
    figure()
    plot3(x_coil, y_coil, z_coil);
    hold on
    quiver3(X, Y, Z, BX_biot_savart, BY_biot_savart, BZ_biot_savart, 'MaxHeadSize', 2);
    xlabel('X (m)');
    ylabel('Y (m)');
    zlabel('Z (m)');
    title('Magnetic Field Vector in the plane (Biot-Savart)')
```

```
68
69      figure()
70      surf(X, Y, Z, B_magnitude);
71      hold on
72      plot3(x_coil, y_coil, z_coil)
73      shading interp;
74      title('Magnetic Field')
75      xlabel('X (m)');
76      ylabel('Y (m)');
77      zlabel('Z (m)');
78      a=colorbar;
79      a.Label.String = 'Magnetic field (T)';
80  end
81
82  end
```

BS-plane.m

Function to calculate magnetic field on a plane with Biot-Savart law with dipoles

```
1  function [BX_dipole, BY_dipole, BZ_dipole, B_magnitude_dipole] = BS_dipole(x_point_plane,
       y_point_plane, x_dipoles, y_dipoles, z_dipoles, ...
2      dipole_moments, X, Y, Z, u0, fig)
3
4  %{
5
6  %%% INPUT %%%
7  x_point_plane: Number of points for x side in the plane
8  y_point_plane: Number of points for y side in the plane
9  x_dipoles: X coordinates of dipoles
10 y_dipoles: Y coordinates of dipoles
11 z_dipoles: Z coordinates of dipoles
12 dipole_moments: dipole moments calculated by minimum norm least-squares solution to linear
       equation
13 X: X coordinates of plane in with you want to calculate magnetic field
14 Y: Y coordinates of plane in with you want to calculate magnetic field
15 Z: Z coordinates of plane in with you want to calculate magnetic field
16 u0: permeability
17 fig: logical value for figures visualization
18
19 %%% OUTPUT %%%
20 Magnetic field and its componentes calculated in the plane [X, Y, Z]
21 BX_dipole: X component of magnetic field calculated with dipole
22 approximation
23 BY_dipole: Y component of magnetic field calculated with dipole
24 approximation
25 BZ_dipole: Z component of magnetic field calculated with dipole
26 approximation
27 B_magnitude_dipole: magnetic field calculated with dipole approximation
28
29 %}
30
31 % Define constant
32 cost_dip = u0/(4*pi);
33
34 BX_dipole = zeros(y_point_plane, x_point_plane);
35 BY_dipole = zeros(y_point_plane, x_point_plane);
36 BZ_dipole = zeros(y_point_plane, x_point_plane);
37
```

```
for k = 1:x_point_plane
    for j = 1:y_point_plane
        Bx = 0;
        By = 0;
        Bz = 0;
        d = 1;

        r_vec = [X(j, k) - x_dipoles(:), Y(j, k) - y_dipoles(:), Z(j, k) - z_dipoles(:)];
        R = sqrt(sum(r_vec.^2, 2));
        non_zero_indices = R ~= 0;
        r_vec = r_vec(non_zero_indices, :);
        R = R(non_zero_indices);
        r_hat = r_vec ./ R;
        m_dot_r_hat = dot(dipole_moments, r_hat, 2);
        dB = cost_dip * (3 .* m_dot_r_hat .* r_hat - dipole_moments) ./ (R.^3);

        BX_dipole(j, k) = sum(dB(:, 1));
        BY_dipole(j, k) = sum(dB(:, 2));
        BZ_dipole(j, k) = sum(dB(:, 3));
    end
end

%Combine magnetic field components into magnitude
B_magnitude_dipole = sqrt(BX_dipole.^2 + BY_dipole.^2 + BZ_dipole.^2);

if fig
    figure()
    plot3(x_dipoles, y_dipoles, z_dipoles, '*')
    hold on
    quiver3(X, Y, Z, BX_dipole, BY_dipole, BZ_dipole);
    xlabel('X (m)');
    ylabel('Y (m)');
    zlabel('Z (m)');
    title('Magnetic Field Vector in the plane (Dipole Approximation)')

    figure()
    surf(X, Y, Z, B_magnitude_dipole);
    hold on
    plot3(x_dipoles, y_dipoles, z_dipoles, '*')
    shading interp;
    title('Magnetic Field with dipoles')
    xlabel('X (m)');
    ylabel('Y (m)');
    zlabel('Z (m)');
    a=colorbar;
    a.Label.String = 'Magnetic field (T)';

    figure()
    surf(X, Y, BX_dipole);
    shading interp;
    hold on
    plot3(x_dipoles, y_dipoles, z_dipoles, '*')
    title('Magnetic Field B_x Component in the Plane (Dipoles)')
    xlabel('X (m)');
    ylabel('Y (m)');
    zlabel('Z (m)');
    a=colorbar;
    a.Label.String = 'Magnetic component x (T)';
```

```
96
97      figure()
98      surf(X, Y, BY_dipole);
99      shading interp;
100     hold on
101     plot3(x_dipoles, y_dipoles, z_dipoles, '*')
102     title('Magnetic Field B_y Component in the Plane (Dipoles)')
103     xlabel('X (m)');
104     ylabel('Y (m)');
105     zlabel('Z (m)');
106     a=colorbar;
107     a.Label.String = 'Magnetic component y (T)';
108
109     figure()
110     surf(X, Y, BZ_dipole);
111     shading interp;
112     hold on
113     plot3(x_dipoles, y_dipoles, z_dipoles, '*')
114     title('Magnetic Field B_z Component in the Plane (Dipoles)')
115     xlabel('X (m)');
116     ylabel('Y (m)');
117     zlabel('Z (m)');
118     a=colorbar;
119     a.Label.String = 'Magnetic component z (T)';
120 end
121
122 end
```

BS-dipole.m

Function to check symmetry on magnetic field calculate with Biot-Savart law with line elements

```
1  function [is_symmetric_Bx, is_symmetric_By, is_symmetric_Bz] = checkFieldSymmetry(x_range,
       y_range, BX_biot_savart, BY_biot_savart, BZ_biot_savart, fig, ...
2      X, Y, x_coil, y_coil, z_coil)
3
4  % Check symmetry for Bx, By, and Bz
5  is_symmetric_Bx = checkSymmetry(BX_biot_savart, x_range);
6  is_symmetric_By = checkSymmetry(BY_biot_savart, x_range);
7  is_symmetric_Bz = checkSymmetry(BZ_biot_savart, x_range);
8
9  if fig
10     figure()
11     surf(X, Y, BX_biot_savart);
12     shading interp;
13     hold on
14     plot3(x_coil, y_coil, z_coil)
15     title('Magnetic Field B_x Component in the Plane (Biot-Savart)')
16     colorbar
17
18     figure()
19     surf(X, Y, BY_biot_savart);
20     shading interp;
21     hold on
22     plot3(x_coil, y_coil, z_coil)
23     title('Magnetic Field B_y Component in the Plane (Biot-Savart)')
24     colorbar
25
26     figure()
```

```
    surf(X, Y, BZ_biot_savart);
    shading interp;
    hold on
    plot3(x_coil, y_coil, z_coil)
    title('Magnetic Field B_z Component in the Plane (Biot-Savart)')
    colorbar
end

end
```

checkFieldSymmetry.m

```
function is_symmetric = checkSymmetry(B_component, range)
    % Check symmetry of the given magnetic field component matrix B_component
    is_symmetric = true;
    mid_idx = ceil(length(range) / 2);

    for i = 1:mid_idx
        % Compare left and right sides of the axis
        if any(abs(B_component(:, i) - B_component(:, end-i+1)) > 1e-6)
            is_symmetric = false;
            disp('The matrix is not symmetric');
            break;
        end
    end
end
```

checkSymmetry.m

Main code

```
close all
clear
clc

%% Variable parameters

% Define point in the plane
x_point_plane = 20; % Number of points for x side in the plane
y_point_plane = 25; % Number of points for y side in the plane
n_plane = x_point_plane*y_point_plane; % Number of point in the plane
plane_z = 0.30; % Height of the parallel plane to 8-figure coil

% Define the number of dipoles
x_dip = 10; % Number of dipoles for x side
y_dip = 20; % Number of dipoles for y side
n_dipoles = x_dip*y_dip; % Number of dipoles

% Define whitch dipole approximation you want to compute
% Using Gauss-Lengendre distribution impose GL = 1
% Using Gauss-Chebyshev distribution impose GL differents from 0 and 1
% Using equidistant nodes impose GL = 0
GL = 1;
% If you choose equidistant nodes, you can choose distance from the center
% of the plane: dist = 0 indicates point farther from the center, dist =
% 0.03 indicates point nearer to the center
dist = 0;

% For visualize all figures impose fig = 1
fig = 0;
```

```
30
31 % Impose it to 1 if you want a complete 8-figure coil with the two straght
32 % lines
33 complete = 1;
34
35 fprintf('Number of dipoles: %d\nPoint in the plane: %d\n', n_dipoles, n_plane)
36
37 %% Create 8-figure coil
38
39 radius_ext = 0.05; % Define the radius external of the spiral
40 radius_in = 0.032; % Define the radius internal of the spiral
41 c = 0.002; % Incerement per revolution
42 center_dist = 0.06; % Distance from the center
43 n_pts_l1 = 3; % Number of point in the line between the two spiral
44 n_pts_l2 = 5; % Number of point in the two straight lines
45
46 [x_coil, y_coil, z_coil, n_segments] = create_8_figure_coil(radius_ext, radius_in, c,
       center_dist, ...
47     n_pts_l1, n_pts_l2, complete);
48
49 fprintf('Number of segments in the coil: %d\n', n_segments)
50
51 if fig
52     figure()
53     plot(x_coil,y_coil)
54     axis equal
55     title('8-figure coil')
56     xlabel('X (m)')
57     ylabel('Y (m)')
58 end
59
60 %% Biot-Savart law for a point
61
62 % Define point P
63 P = [0.05, 0, 0.25];
64
65 % Calculate differential length elements
66 deltal = [diff(x_coil); diff(y_coil); diff(z_coil)]';
67
68 % Calculate the midpoints of each segment
69 midpoints = (([x_coil(1:end-1); y_coil(1:end-1); z_coil(1:end-1)] + ...
70               [x_coil(2:end); y_coil(2:end); z_coil(2:end)] ) ./ 2)';
71
72 % Define constant current and permeability
73 I = 1;
74 u0 = 4*pi*10^(-7);
75
76 [B_point] = BS_one_point(P, deltal, midpoints, I, u0, fig, x_coil, y_coil, z_coil);
77
78 %% Biot-Savart law for a plane
79
80 % Define a plane
81 x_range = linspace(-10*radius_ext, 10*radius_ext, x_point_plane);
82 y_range = linspace(-10*radius_ext, 10*radius_ext, y_point_plane);
83 [X, Y] = meshgrid(x_range, y_range);
84 Z = plane_z*ones(size(Y));
85
86 if fig
```

```
    figure()
    plot3(x_coil, y_coil, z_coil);
    hold on
    plot3(X, Y, Z, '*');
    title('Position of plane and 8-figure coil')
    xlabel('X (m)');
    ylabel('Y (m)');
    zlabel('Z (m)');
end

[BX_biot_savart, BY_biot_savart, BZ_biot_savart, B_magnitude] = BS_plane(x_point_plane,
    y_point_plane, X, Y, Z, ...
    deltal, midpoints, I, u0, fig, x_coil, y_coil, z_coil);

%% Check the simmetry of the field components

[is_symmetric_Bx, is_symmetric_By, is_symmetric_Bz] = checkFieldSymmetry(x_range, y_range,
    ...
    BX_biot_savart, BY_biot_savart, BZ_biot_savart, fig, X, Y, x_coil, y_coil, z_coil);

% Display the results
fprintf('Symmetry check results:\n');
fprintf('B_x is symmetric: %d\n', is_symmetric_Bx);
fprintf('B_y is symmetric: %d\n', is_symmetric_By);
fprintf('B_z is symmetric: %d\n', is_symmetric_Bz);

%% Create dipole plane

x_min = -2*radius_ext;
x_max = 2*radius_ext;
y_min = -radius_ext;
y_max = radius_ext;

if GL == 1
    % Using Gauss-Lengendre distribution
    [x_nodes, x_weights] = lgwt(x_dip, x_min, x_max);
    [y_nodes, y_weights] = lgwt(y_dip, y_min, y_max);
elseif GL == 0
    % Using equidistant nodes
    x_nodes = linspace(x_min+dist, x_max-dist, x_dip);
    y_nodes = linspace(y_min+dist/2, y_max-dist/2, y_dip);
else
    % Using Gauss-Chebyshev distribution
    theta_x = (2*(1:x_dip) - 1) * pi / (2*x_dip);
    theta_y = (2*(1:y_dip) - 1) * pi / (2*y_dip);
    x_nodes = cos(theta_x).*0.075;
    y_nodes = cos(theta_y).*0.05;
end

[x_dipoles, y_dipoles] = meshgrid(x_nodes, y_nodes);

plane_z_dip = 0;
z_dipoles = plane_z_dip*ones(size(y_dipoles));

if fig
    figure
    plot3(x_dipoles, y_dipoles, z_dipoles, '*')
    hold on
```

```
    plot3(x_coil,y_coil, z_coil)
    title('Dipole coil')
    xlabel('X (m)')
    ylabel('Y (m)')
    zlabel('Z (m)')
end

% Assuming dipole moment
dip = [1,1,1];

%% Calculate matrix A

Ax = zeros(n_plane, n_dipoles);
Ay = zeros(n_plane, n_dipoles);
Az = zeros(n_plane, n_dipoles);
d=1;

% Define constant
cost_dip = u0 / (4 * pi);

for n = 1:x_dip
    for l = 1:y_dip
        for k = 1:x_point_plane
            for j = 1:y_point_plane
                r_vec = [X(j, k) - x_dipoles(l, n), Y(j, k) - y_dipoles(l, n), Z(j, k) -
                    z_dipoles(l, n)];
                R = sqrt(sum(r_vec.^2, 2));
                if R == 0
                    continue
                end
                r_hat = r_vec ./ R;
                m_dot_r_hat = dot(dip, r_hat, 2);
                dB = cost_dip * (3 .* m_dot_r_hat .* r_hat - dip) ./ (R.^3);
                a = dB ./ dip;

                idx = j+(k-1)*y_point_plane;
                Ax(idx,d) = Ax(idx,d) + a(1);
                Ay(idx,d) = Ay(idx,d) + a(2);
                Az(idx,d) = Az(idx,d) + a(3);
            end
        end
        d = d+1;
    end
end

%% Calculate real dipole moments

% Flatten the magnetic field components into a single vector B
B = zeros(n_plane, 3);

for k = 1:x_point_plane
    for j = 1:y_point_plane
        idx = j+(k-1)*y_point_plane;
        B(idx,1) = BX_biot_savart(j,k);
        B(idx,2) = BY_biot_savart(j,k);
        B(idx,3) = BZ_biot_savart(j,k);
    end
end
```

```
200
201 max_iterations = min(size(Ax,1),20);
202 tolerance = 1e-6;
203
204 if n_plane >= n_dipoles
205     fprintf('Least-squares method is used\n');
206     % Check if least-squares solution converges
207     while true
208         % Using least-squares solution to linear equations
209         [mom_x, flag_x] = lsqr(Ax, B(:,1), tolerance, max_iterations);
210         [mom_y, flag_y] = lsqr(Ay, B(:,2), tolerance, max_iterations);
211         [mom_z, flag_z] = lsqr(Az, B(:,3), tolerance, max_iterations);
212
213         % Check if all flags are zero
214         if flag_x == 0 && flag_y == 0 && flag_z == 0
215             break;
216         end
217
218         % Increase the number of iterations for the next run
219         max_iterations = max_iterations + 10;
220     end
221 elseif n_plane < n_dipoles
222     fprintf('Minimum norm least-squares method is used\n');
223     % Using minimum norm least-squares solution to linear equation
224     lambda = 1e-4;
225     Ax_reg = Ax + lambda .* eye(size(Ax));
226     Ay_reg = Ay + lambda .* eye(size(Ay));
227     Az_reg = Az + lambda .* eye(size(Az));
228
229     mom_x = lsqminnorm(Ax_reg, B(:,1), 'warn');
230     mom_y = lsqminnorm(Ay_reg, B(:,2), 'warn');
231     mom_z = lsqminnorm(Az_reg, B(:,3), 'warn');
232 end
233
234 % Real dipole moments
235 dipole_moments = [mom_x, mom_y, mom_z];
236
237 if fig
238     figure()
239     quiver3(x_dipoles(:), y_dipoles(:), z_dipoles(:), mom_x, mom_y, mom_z);
240     xlabel('X (m)');
241     ylabel('Y (m)');
242     zlabel('Z (m)');
243     title('Magnetic Dipoles');
244
245     figure()
246     plot3(x_dipoles, y_dipoles, z_dipoles, '*');
247     xlabel('X (m)');
248     ylabel('Y (m)');
249     zlabel('Z (m)');
250     title('Dipole coil');
251 end
252
253 %% Biot-Savart law with dipole approximation
254
255 [BX_dipole, BY_dipole, BZ_dipole, B_magnitude_dipole] = BS_dipole(x_point_plane,
        y_point_plane, x_dipoles, y_dipoles, z_dipoles, ...
256     dipole_moments, X, Y, Z, u0, fig);
```

```
257
258 %% Comparison between two methods
259
260 % Calculate mean square error
261 MSE_BX = immse(BX_biot_savart, BX_dipole);
262 MSE_BY = immse(BY_biot_savart, BY_dipole);
263 MSE_BZ = immse(BZ_biot_savart, BZ_dipole);
264 MSE_B = immse(B_magnitude, B_magnitude_dipole);
265
266 % Calculate MAE
267 Abs_B = mean(abs(B_magnitude - B_magnitude_dipole), 'all');
268
269 % Calcualte Relative E
270 Rel_Error = mean(abs((B_magnitude - B_magnitude_dipole) ./ B_magnitude), 'all');
271 Percent_B = Rel_Error * 100;
272
273 % Create a table with the calculated MSE values
274 MSE_Table = table(MSE_BX, MSE_BY, MSE_BZ, MSE_B, Abs_B, Percent_B, ...
275     'VariableNames', {'MSE_BX', 'MSE_BY', 'MSE_BZ', 'MSE_B', 'Abs_B', 'Percent_B'});
276
277 disp(MSE_Table);
```

main.m

Function for plotting MAE and relative error of magnetic field **B** for all combinations of points in the plane and number of dipoles for each distribution

```
1  function [] = plot_E_distribution(tables, names, idx)
2  %{
3  PLOTS_TABLES generates heatmaps for each distribution with
4  n_dipoles on the x-axis and n_plane on the y-axis.
5
6  %%% INPUT %%%
7  tables: cell array of tables containing error data
8  names: cell array of names for legend entries
9  idx: index of the column to plot from each table
10 %}
11
12 % Loop through each table to create a heatmap
13 for i = 1:length(tables)
14     tbl = tables{i};
15
16     % Extract unique values of n_plane and n_dipoles
17     n_planes = unique(tbl.n_plane);
18     n_dipoles = unique(tbl.n_dipoles);
19
20     % Initialize the matrix to store the values for the heatmap
21     heatmap_data = NaN(length(n_planes), length(n_dipoles));
22
23     % Fill the heatmap matrix with the data
24     for j = 1:length(n_planes)
25         for k = 1:length(n_dipoles)
26             % Find the corresponding rows in the table
27             rows = tbl(tbl.n_plane == n_planes(j) & tbl.n_dipoles == n_dipoles(k), :);
28             if ~isempty(rows)
29                 % If there are multiple rows, take the mean value
30                 heatmap_data(j, k) = mean(rows{:, idx});
31             end
32         end
```

```
    end

    % Create a new figure for the heatmap
    figure();

    % Plot the heatmap
    imagesc(heatmap_data);
    c = colorbar;

    % Customize the axes
    xticks(1:length(n_dipoles));
    xticklabels(n_dipoles);
    xlabel('Number of dipoles');

    yticks(1:length(n_planes));
    yticklabels(n_planes);
    ylabel('Number of points in the plane');


    % Customize the colorbar label based on the index
    if idx == 8
        c.Label.String = 'Relative error (%)';
        str_fig = sprintf('%s_heatmap_E.png', names{i});
    else
        c.Label.String = 'MAE (T)';
        str_fig = sprintf('%s_heatmap_A.png', names{i});
    end

    % Add a title
    title(names{i});

    % Save the figure
    saveas(gcf, fullfile('heatmap', str_fig));

end

end
```

plot-E-distribution.m

Function for plotting MAE and relative error of magnetic field **B** for various numbers of dipoles and distributions, given a specified number of points in the plane

```
function [] = plot_E(tables, names, str, idx, n_plane_filter, flag)
%{
PLOTS_TABLES plots absolute E, realtive E for all combination of points in the plane and
dipoles

%%% INPUT %%%
tables: cell array of tables containing error data
names: cell array of names for legend entries
str: title of plot
idx: index of the column to plot from each table
n_plane_filter: value of n_plane to filter the rows to be plotted
flag: 0 for minimum norm least squares method;
      1 for least squares method;
      2 for both
%}

```

```
% Create a new figure
figure();
hold on;

% Loop through each table and plot the data
for i = 1:length(tables)
    tbl = tables{1, i};

    % Filter rows based on the value of n_plane
    filtered_tbl = tbl(tbl.n_plane == n_plane_filter, :);

    if isempty(filtered_tbl)
        continue;
    end

    % Additional filtering based on the flag
    if flag == 0
        filtered_tbl = filtered_tbl(filtered_tbl{:, 1} < filtered_tbl{:, 2}, :);
    elseif flag == 1
        filtered_tbl = filtered_tbl(filtered_tbl{:, 1} > filtered_tbl{:, 2}, :);
    elseif flag ~= 2
        error('Invalid flag value. Use 0 for first < second, 1 for first > second, or 2 for
            all rows.');
    end

    if isempty(filtered_tbl)
        continue;
    end

    x1 = filtered_tbl{:, 1};
    x2 = filtered_tbl{:, 2};
    y = filtered_tbl{:, idx};

    % Combine the first two columns into a single set of labels
    x_labels = strcat(num2str(x2));

    % Plot the data
    plot(y, 'o-', 'DisplayName', names{i});

    % Customize the x-axis with combined labels
    xticks(1:length(x_labels));
    xticklabels(x_labels);
end

xlabel('number of dipoles');
if idx == 8
    ylabel('(%)')
    if flag == 0
        str_fig = sprintf('%d_mlsm_E.png', n_plane_filter);
    elseif flag == 1
        str_fig = sprintf('%d_lsm_E.png', n_plane_filter);
    else
       str_fig = sprintf('%d_all_E.png', n_plane_filter);
    end
else
    ylabel('(T)')
    if flag == 0
        str_fig = sprintf('%d_mlsm_A.png', n_plane_filter);
```

```
    elseif flag == 1
        str_fig = sprintf('%d_lsm_A.png', n_plane_filter);
    else
       str_fig = sprintf('%d_all_A.png', n_plane_filter);
    end
end
title(str);
legend('show');

grid on;
hold off;

saveas(gcf, fullfile('plot', str_fig));
```

plot-E.m

Function for plotting MAE and relative error of magnetic field **B** for various numbers of dipoles and distributions, given a specified number of points in the plane using heatmap

```
function [] = plot_E_heatmap(tables, names, str, idx, n_plane_filter, flag)
%{
PLOTS_TABLES plots absolute E, realtive E for all combination of points in the plane and
dipoles

%%% INPUT %%%
tables: cell array of tables containing error data
names: cell array of names for legend entries
str: title of plot
idx: index of the column to plot from each table
n_plane_filter: value of n_plane to filter the rows to be plotted
flag: 0 for minimum norm least squares method;
      1 for least squares method;
      2 for both
%}

% Create a new figure
figure();
hold on;

% Initialize data arrays for heatmap
heatmap_data = [];
x_labels = [];
y_labels = {};

% Loop through each table and gather data for heatmap
for i = 1:length(tables)
    tbl = tables{1, i};

    % Filter rows based on the value of n_plane
    filtered_tbl = tbl(tbl.n_plane == n_plane_filter, :);

    if isempty(filtered_tbl)
        continue;
    end

    % Additional filtering based on the flag
    if flag == 0
        filtered_tbl = filtered_tbl(filtered_tbl{:, 1} < filtered_tbl{:, 2}, :);
    elseif flag == 1
```

```
        filtered_tbl = filtered_tbl(filtered_tbl{:, 1} > filtered_tbl{:, 2}, :);
    elseif flag ~= 2
        error('Invalid flag value. Use 0 for first < second, 1 for first > second, or 2 for
            all rows.');
    end

    if isempty(filtered_tbl)
        continue;
    end

    x1 = filtered_tbl{:, 1};
    x2 = filtered_tbl{:, 2};
    y = filtered_tbl{:, idx};

    % Combine the first two columns into a single set of labels
    x_labels = [x_labels; strcat(num2str(x2))]; % Collecting labels

    % Collect data for heatmap
    heatmap_data = [heatmap_data; y'];
    y_labels{end+1} = names{i};
end

% Convert collected data to appropriate format for heatmap
%heatmap_data = cell2mat(heatmap_data);

% Generate the heatmap
imagesc(heatmap_data);
colorbar;

% Customize the axes with combined labels
xticks(1:size(heatmap_data, 2));
xticklabels(x_labels);
yticks(1:length(y_labels));
yticklabels(y_labels);

xlabel('number of dipoles');
if idx == 8
    ylabel('(%)')
    if flag == 0
        str_fig = sprintf('%d_mlsm_E.png', n_plane_filter);
    elseif flag == 1
        str_fig = sprintf('%d_lsm_E.png', n_plane_filter);
    else
       str_fig = sprintf('%d_all_E.png', n_plane_filter);
    end
else
    ylabel('(T)')
    if flag == 0
        str_fig = sprintf('%d_mlsm_A.png', n_plane_filter);
    elseif flag == 1
        str_fig = sprintf('%d_lsm_A.png', n_plane_filter);
    else
       str_fig = sprintf('%d_all_A.png', n_plane_filter);
    end
end
title(str);
hold off;

```

```
saveas(gcf, fullfile('heatmap', str_fig));

end
```

plot-E-heatmap.m

Functions for plotting mean squared error of magnetic field components for all combinations of points in the plane and number of dipoles

```
function [] = plots_MSE(tables, names, idx, n_plane_filter, flag)
%{
PLOTS_3COMPONENTS plots MSE of the magnetic field components from multiple
tables when magnetic field is calculated with three components

%%% INPUT %%%
tables: cell array of tables containing MSE data
names: cell array of names for legend entries
idx: index of the row to plot from each table, corresponding to a specific
combination of number of points in the plane and number of dipoles
n_plane_filter: value of n_plane to filter the rows to be plotted
flag: 0 to plot rows where the first column is less than the second column;
      1 to plot rows where the first column is greater than the second column;
      2 to plot all rows
%}

% Define the names for x-axis labels
nam = {'x', 'y', 'z', 'magnitude'};

% Specify columns to plot which are stored magnetic field components
columns = 3:6;

hold on;

% Loop through each table in tables
for i = 1:length(tables)
    table_data = tables{i};

    % Filter rows based on the value of n_plane
    filtered_tbl = table_data(table_data.n_plane == n_plane_filter, :);

    % Additional filtering based on the flag
    if flag == 0
        filtered_tbl = filtered_tbl(filtered_tbl{:, 1} < filtered_tbl{:, 2}, :);
    elseif flag == 1
        filtered_tbl = filtered_tbl(filtered_tbl{:, 1} > filtered_tbl{:, 2}, :);
    elseif flag ~= 2
        error('Invalid flag value. Use 0 for first < second, 1 for first > second, or 2 for
            all rows.');
    end

    % Ensure the idx is within the bounds of the filtered table
    if idx > height(filtered_tbl)
        warning('Index exceeds the number of rows in the filtered table for dataset %d.',
            i);
        continue;
    end

    % Extract data from the table for the specified row
    data_to_plot = filtered_tbl{idx, columns};
```

```
49
50      % Plot the data
51      plot(data_to_plot, 'o-', 'DisplayName', names{i});
52
53      xticks(1:length(columns));
54      xticklabels(nam);
55  end
56
57  xlabel('Magnetic field components');
58  str = sprintf('MSE: %d points in the plane', n_plane_filter);
59  ylabel(str);
60  title(sprintf('%d dipoles', filtered_tbl{idx, 2}));
61  grid on;
62  legend('show');
63  hold off;
64
65  end
```

plot-MSE.m

Code for visualizing figures of relative error and mean squared error

```
1  clear
2  close all
3  clc
4
5  %% Show graphs
6
7  % Read the data from the CSV files
8  eq_0 = readtable("Equidistant_0.csv");
9  eq_003 = readtable("Equidistant_003.csv");
10 GC = readtable("Gauss_Chebyshev.csv");
11 GL = readtable("Gauss_Lengendre.csv");
12
13 tables = {eq_0, eq_003, GC, GL};
14 names = {'eq0', 'eq003', 'GC', 'GL'};
15 % 300, 400, 450, 500, 600 and 750 points
16 n_plane = 400;
17
18 % flag: 0 for minimum norm least squares method;
19 % 1 for least squares method;
20 % 2 for both
21 flag = 2;
22
23 str_pe = sprintf('Relative E: %d points in the plane', n_plane);
24 str_ae = sprintf('MAE: %d points in the plane', n_plane);
25
26 % Show heatmap: number of dipoles - number on plane for each distribution
27 plot_E_distribution(tables, names, 7)
28 plot_E_distribution(tables, names, 8)
29
30 % Show heatmap: number of dipoles - four distribution for a give plane
31 plot_E_heatmap(tables, names, str_pe, 8, n_plane, flag)
32 plot_E_heatmap(tables, names, str_ae, 7, n_plane, flag)
33
34 % Show plot: number of dipoles - four distribution for a give plane
35 plot_E(tables, names, str_pe, 8, n_plane, flag);
36 plot_E(tables, names, str_ae, 7, n_plane, flag);
37
```

```
% Show MSE
if flag == 2
    figure
    for i = 1:12
        subplot(3, 5, i)
        plots_MSE(tables, names, i, n_plane, flag)
    end
end
```

graph.m