Master thesis

Philip Mark Suskin

Optimal Control of a Field-Free-Point Using a Multi-Coil Magnetic Field Generator

April 14, 2025

supervised by:
Prof. Dr.-Ing. Tobias Knopp
Dr. rer. nat. Martin Möddel
Fynn Förger

Hamburg University of Technology
Institute for Biomedical Imaging
Schwarzenbergstraße 95
21073 Hamburg

University Medical Center Hamburg-Eppendorf
Section for Biomedical Imaging
Martinistraße 52
20246 Hamburg

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt zu haben.

Hamburg, den 11.04.2024

Contents

1 Introduction

Dynamic field generation is an essential but highly power-consuming process. In particular, as systems are scaled up in size, generating associated magnetic fields becomes a highly arduous, if not infeasible, task [1]. This thesis investigates the control of multi-coil systems to optimize power efficiency, with a particular focus on modeling the non-linear current-to-field relationship introduced by the use of soft iron cores in the field generator's coil design. Using optimization techniques and neural networks, a novel method for controlling generated magnetic fields in a power-efficient manner is proposed.

The demonstrations in this thesis are based on experiments using a prototypical selection field generator intended for Magnetic Particle Imaging. Magnetic Particle Imaging (MPI) is a tomographic medical imaging method based on the determination of iron oxide nanoparticle concentration [2]. The magnetic fields required for MPI are generated through complex multi-coil systems and are separated into a selection field and a focus field, which play distinct roles in spatial encoding and signal generation, respectively. Of the two fields, the selection field requires the most power, presenting a challenge as MPI systems are scaled to human-sized applications.

Selection fields facilitate spatial encoding by producing a Field-Free Point (FFP) or Field-Free Line (FFL). These are crucial to MPI, as they define regions in which the magnetic field strength is approximately zero, enabling the detection of nanoparticles by measuring the dynamic response to shifts in their position. To this end, for imaging, the position of either an FFP or FFL is shifted over time along some defined trajectory while superimposed by a respective gradient strength. The series of fields that must be generated in order to produce FFPs/FFLs along the defined trajectory is called a field sequence.

The selection field generator used for experimentation comprises 18 total coils arranged in two parallel 3×3 coil grids spaced 10 cm apart, see Figure 1.1. Within each grid, coils are positioned side by side with a center-to-center distance of 5 cm between adjacent coils. These coils are (independently) powered by currents, leading to the generation of magnetic fields with an FFP in the field space.

Magnetic fields can be represented by an expansion based on spherical harmonics [3]. Spherical harmonics form a basis for source-free quasi-static magnetic fields and thus provide efficient representations of magnetic fields within some defined sphere radius $R \in \mathbb{R}^+$.

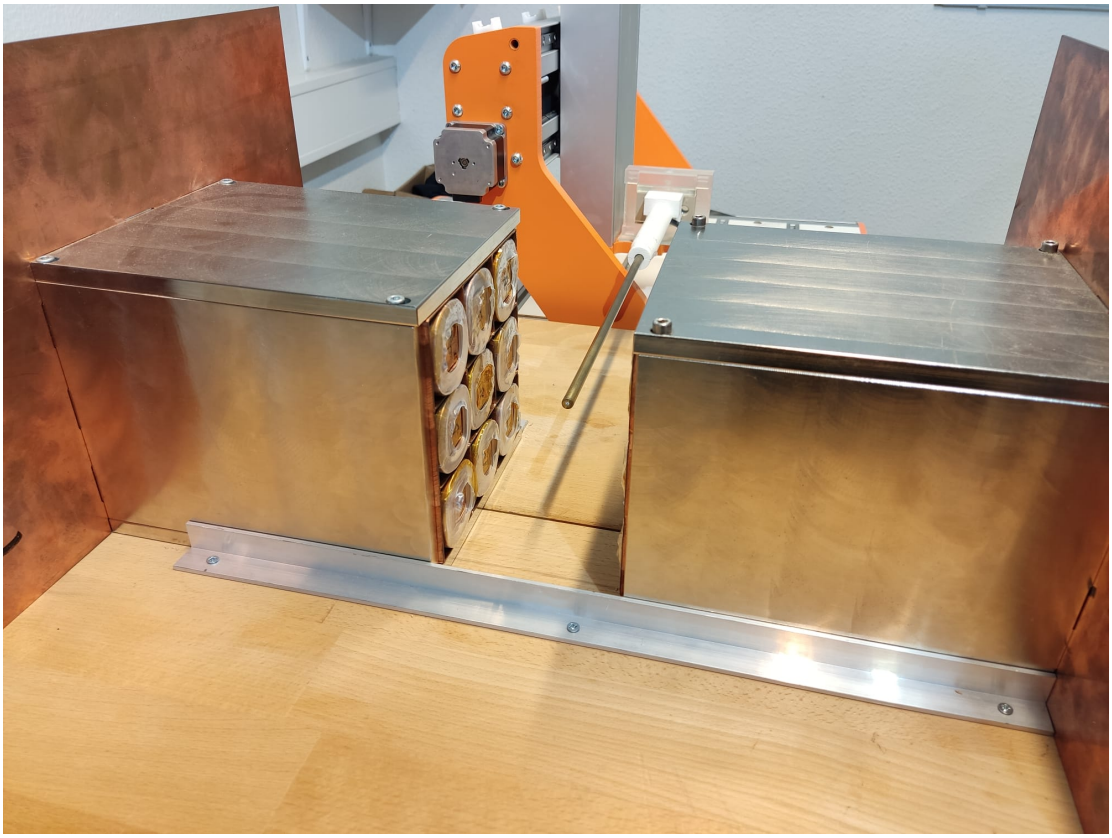


Figure 1.1: MPI selection field generator setup

Using the proposed setup to establish a proof of concept, the main objective in this work involves determining an optimal[1] current sequence that produces some target field sequence, a variation of the *inverse current problem*. The goal is to show that this can be achieved by modeling the *forward problem* (current sequence to field sequence) with neural-network-based approximations around spherical harmonics and subsequently optimizing over the current sequence w.r.t. power consumption within the constraints posed by the field sequence.

[1] An optimal current sequence is one that is minimal w.r.t. the sum of the squares of the individual current strengths, since this is proportional to power consumption.

2 Problem Statement

The selection field generator used for experimentation comprises 18 soft iron core coils arranged to be capable of producing desired magnetic fields. However, this system is intrinsically integrated with a non-linear relationship between current and magnetic field, i.e., changes to input currents are not proportional to resulting changes to the output magnetic field. Consequently, the key challenge of this work lies in developing a control algorithm that is capable of modeling this non-linear relationship, while also minimizing power consumption and meeting constraints stemming from both the desired magnetic field sequence as well as the hardware in medical imaging applications.

In the context of the detailed FFP-based selection field generator, the primary goal of this work involves designing an algorithm that effectively controls the positioning of an FFP over time, which is crucial for spatial encoding in MPI. This is achieved through optimization of current sequences applied to the multi-coil system. Power consumption is minimized under constraints on field properties, such as the precision of FFP position within a defined trajectory, as well as constraints pertaining to the hardware, i.e., to the currents applied to each coil. These constraints must reflect both theoretical considerations and practical limitations based on real-world measurements and system specifications.

The challenges associated with the defined optimization problem include the following:

1. Accurately modeling the non-linear current-to-field relationship integrated into the selection field generator.
2. Designing a control algorithm for field sequences (FFP trajectories) that minimizes power consumption while maintaining the required field dynamics.
3. Implementing a scalable, computationally efficient approach that remains feasible for real-time medical imaging applications when scaling up bore size[1].

This thesis aims to provide a proof of concept for an approach which addresses these challenges by experimenting on a simplified model of the multi-coil field generator. The experimental setup is illustrated in Figure 2.1. In this setup, only the 4 coils in the top-left corner of the left grid are powered by current. In Section 3.1.3, a method for simulatively augmenting this setup is detailed, such that the four coils on the right grid which are positioned directly opposite the initial four coils, are treated as if they were powered by current as well.

[1] In the context medical imaging applications, bore size refers to the diameter of the cylindrical space in which a patient lies.

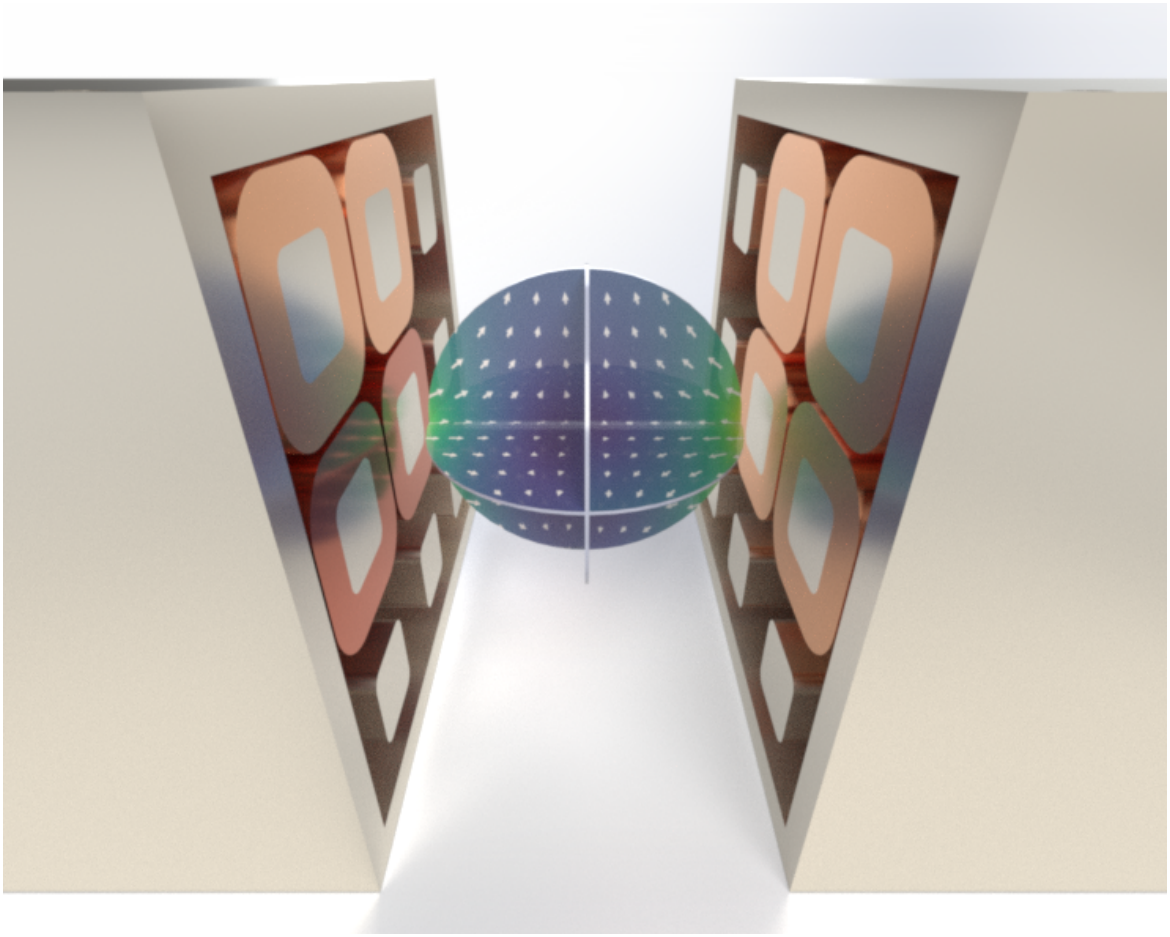


Figure 2.1: Visualization of experimental setup

3

Experiments & Methods

In order to achieve power-efficient control of field sequences for the presented selection field generator, a multi-faceted algorithm based on neural networks and non-linear optimization was developed. A visual representation of the algorithmic approach is presented in Figure 3.1, wherein the key components of the surrogate model for the forward problem, field estimation and singular value estimation, are visualized in their neural network form. This section outlines the experiments, design choices, and methodology used to model the examined system's non-linear current-to-field relationship, optimize its power consumption, and evaluate the resulting control algorithm.

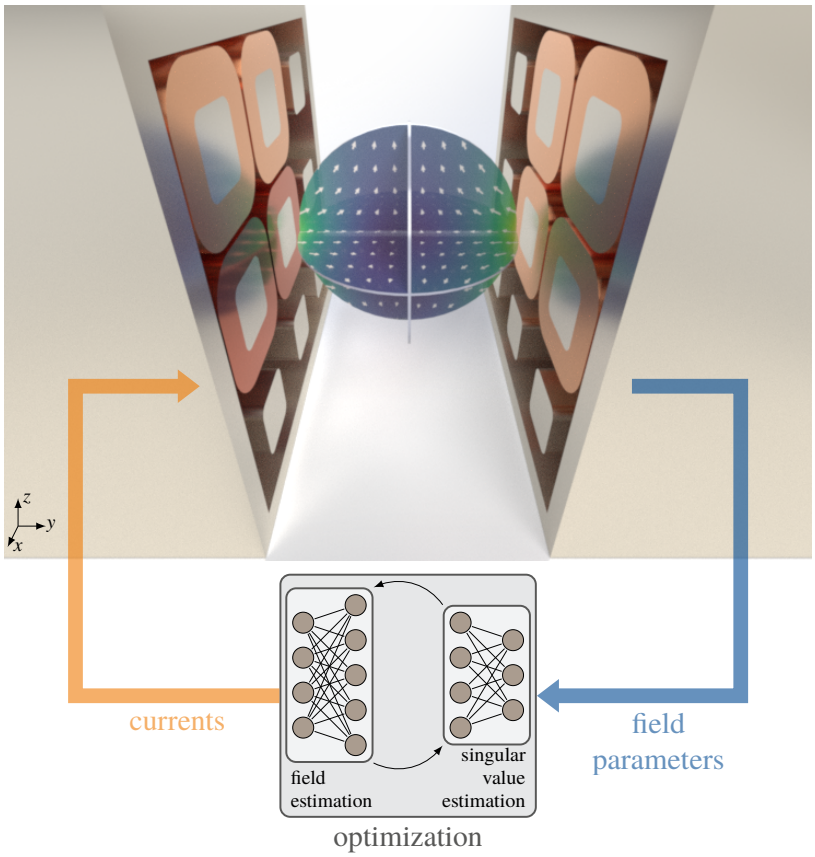


Figure 3.1: Visualization of optimization approach

3.1 Current Sequence Optimization

3.1.1 Problem Definition

The methodology of this thesis formalizes the task of optimal control as an optimization problem, where the input variables are the current sequences $\boldsymbol{I}(t)\colon [0,T] \mapsto \mathbb{R}^3,\ T \in \mathbb{R}$ applied to each of the four coils in the top left of the left grid of the selection field generator (mirrored onto the four coils in the top left of the right grid). The objective of the optimization is to minimize power consumption $P(\cdot)$, in this case the sum of the squares of the input currents, while maintaining control over the FFP given some FFP trajectory $\boldsymbol{\gamma}(t)\colon [0,T] \mapsto \mathbb{R}^3,\ T \in \mathbb{R}$, i.e., keeping the magnetic field strength $\boldsymbol{B}\left(\boldsymbol{\gamma}\left(t\right), \boldsymbol{c}\left(\boldsymbol{I}\left(t\right)\right)\right) : \mathbb{R}^3 \times \mathbb{R}^{3\times n} \mapsto \mathbb{R}^3,\ n \in \{x^2 | x \in \mathbb{Z}\}$ close to zero at the target FFP position $\boldsymbol{\gamma}(t)$ based on the spherical harmonic coefficients $\boldsymbol{c}\left(\boldsymbol{I}\left(t\right)\right)$ computed by the field prediction model for each step in the sequence.

The objective function for the optimization problem is defined as follows:

$$\underset{\boldsymbol{I}(t)}{\operatorname{argmin}}\, P\left(\boldsymbol{I}\left(t\right)\right),\ P\left(\boldsymbol{I}\left(t\right)\right) \propto \|\boldsymbol{I}(t)\|_2^2,\ \|\cdot\|_2 \text{ is the } \ell_2 \text{ norm.} \tag{3.1}$$

Constraints to the optimization problem that reflect both physical hardware limitations as well as field dynamic requirements are also defined. The hardware constraints concern current bounds and slew rate[1] limits and are defined as follows:

$$I_{min} \leq I_n(t) \leq I_{max},\ n \in [1,4], \tag{3.2a}$$

$$\Delta I_{min} \leq \dot{I}_n(t) \leq \Delta I_{max},\ n \in [1,4]. \tag{3.2b}$$

The field dynamic constraints concern FFP trajectory precision and are defined as follows:

$$\left|B_i\left(\boldsymbol{\gamma}\left(t\right), \boldsymbol{c}\left(\boldsymbol{I}\left(t\right)\right)\right)\right| \leq \epsilon,\ i \in \{x,y,z\}, \tag{3.3a}$$

$$\sigma_{\min}\left(J_{\boldsymbol{r}}\boldsymbol{B}|_{(\boldsymbol{\gamma}(t),\boldsymbol{c}(\boldsymbol{I}(t)))}\right) \geq g, \tag{3.3b}$$

where $\sigma_{\min}\colon \mathbb{R}^{n\times m} \mapsto \mathbb{R},\ n,m \in \mathbb{N}$ denotes the smallest singular value of some two-dimensional matrix, in this case the Jacobian of the matrix for the magnetic field with respect to position $\boldsymbol{r}$ in three-dimensional field space evaluated at the current FFP position $\boldsymbol{\gamma}(t)$ for the predicted spherical harmonic coefficients $\boldsymbol{c}\left(\boldsymbol{I}\left(t\right)\right)$. It is possible to constrain the gradient field in this way, since the quasi-static approximation of Maxwell's equations is

[1] The slew rate defines the maximum rate of change for an amplifier's output voltage, which is proportional to the current for a fixed resistance.

assumed for the system used in this work. This implies that the following one of Maxwell's equations,

$$\nabla \times \boldsymbol{B} = \mu_0 \left(\boldsymbol{J} + \epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t} \right), \tag{3.4}$$

reduces to zero when no current density $\boldsymbol{J}$ is present, since the time derivative of the E-field $\partial \boldsymbol{E}/\partial t$ can be neglected. This necessarily means that the Jacobian of the magnetic field with respect to position evaluates to a symmetric matrix for which, due to the invariance under unitary transformation of symmetric matrices, the smallest singular value determines the gradient field strength.

Each of the implemented constraints underlies a certain physical explanation for the system used in this work. Equation 3.2a represents a natural range of coil current bounds, in addition to matching the training currents used for the development of a field prediction model, detailed in Section 3.1.3. Equation 3.2b limits the slew rate, i.e., the change in current for each respective coil between each time step, as currents in coils cannot be changed arbitrarily quickly by the magnetic amplifiers in this system. Furthermore, there are physical constraints on the slew rate due to Lenz's law, which states that a changing magnetic field induces an electric current whose direction is such that the magnetic field generated by the induced current opposes changes in the initial magnetic field [4]. Equation 3.3a limits the magnetic field strength at the target FFP position, which ideally equals zero. Equation 3.3b ensures a sufficient gradient field strength at the target FFP position, which, together with Equation 3.3a limits the distance[2] that can lie between the true FFP and the proposed FFP, i.e., limits the *FFP offset*, at any given time step in the sequence. This can be shown using the following simplified equation for the magnetic field at the FFP position:

$$\|\boldsymbol{B}_{\boldsymbol{\gamma}}\| = \|\boldsymbol{g}_{\boldsymbol{\gamma}}\| \cdot \|\Delta \boldsymbol{r}\|, \tag{3.5}$$

where $\boldsymbol{B}_{\boldsymbol{\gamma}}$ and $\boldsymbol{g}_{\boldsymbol{\gamma}}$ are the field strength and gradient strength at a position $\boldsymbol{\gamma}$, respectively, and $\Delta \boldsymbol{r}$ is the FFP offset. Transformed, this equation directly illustrates the relationship between FFP offset and both field strength and gradient strength:

$$\|\Delta \boldsymbol{r}\| = \frac{\|\boldsymbol{B}_{\boldsymbol{\gamma}}\|}{\|\boldsymbol{g}_{\boldsymbol{\gamma}}\|}. \tag{3.6}$$

This allows for estimation of the expected maximum FFP offset based on the constraints defined for $\boldsymbol{B}$ (Equation 3.3a) and $\boldsymbol{g}$ (Equation 3.3b). For example, constraining the magnetic field strength to values between $\pm 1 \times 10^{-4}$ T and the gradient strength to values above 0.1 T/m limits the expected FFP offset to approximately 1×10^{-3} m, with decreases to the magnetic field strength constraint and increases to the gradient strength constraint each leading to a smaller expected maximum FFP offset.

[2] Euclidean distance.

3.1.2 Optimization Algorithm

IPOPT (Interior Point OPTimizer) [5], an open source software package for large-scale non-linear optimization based on first- and second-order derivatives, is used as a solver for the optimization problem defined by these conditions. In general, IPOPT is capable of solving non-linear problems of the following form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & h^L \leq h(x) \leq h^U \\ & x^L \leq x \leq x^U, \end{aligned} \tag{3.7}$$

where $x \in \mathbb{R}^n$ are the optimization variables with lower and upper bounds $x^L \in (\mathbb{R} \cup \{-\infty\})^n$ and $x^U \in (\mathbb{R} \cup \{+\infty\})^n$ with $x^L \leq x^U$, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $h : \mathbb{R}^n \to \mathbb{R}^m$ are the non-linear constraints.

As an interior point method (IPM), IPOPT traverses the interior of the feasible region, i.e. the region of the solution space in which all constraints are satisfied [6]. In order to stay within this region without having to directly handle inequality constraints[3], *barrier functions*, functions which discourage the optimizer from approaching the boundary of the feasible region too closely, are implemented. IPOPT implements a logarithmic barrier function ϕ, which penalizes solutions approaching the boundaries of the constraints $h(x) \geq 0$ (constraints of the form $h(x) \bowtie c,\ c \in \mathbb{R}^m$ can be brought into this form via linear transformation) as follows:

$$\phi(x, \mu) = f(x) - \mu \sum_i \ln(h_i(x)), \tag{3.8}$$

where $\mu \in \mathbb{R}^+$ is the barrier parameter controlling the weight of the penalty. As $\mu \to 0$, the solution of the barrier problem converges to the solution of the original optimization problem. To this end, μ is iteratively reduced between solutions to the barrier problem found using a linear solver, which involves satisfying the *Karush-Kuhn-Tucker (KKT) conditions* [7], a set of conditions necessary for a solution to be optimal in a constrained optimization problem. For each of these conditions, residuals (*KKT residuals*) can be computed to assess convergence toward optimality. The linear solver used as part of IPOPT in the implementation as part of this thesis is MUMPS (MUltifrontal Massively Parallel sparse direct Solver) [8] [9], an algorithm based on the *multifrontal method* [10] which is commonly implemented for solving large and sparse systems of linear equations. An algorithmic definition of IPOPT is provided in Algorithm 3.1.

[3] Indirect handling of inequality constraints has several practical and computational benefits. Inequality constraints introduce issues such as numerical instability along the boundaries of the feasible region, since the lower/upper bounds of inequality constraints lead to non-differentiable points in the solution space.

Algorithm 3.1 Interior Point Method for Nonlinear Optimization

Input: Objective function $f(x)$, inequality constraints $h(x) \geq 0$, initial guess x_0, barrier parameter $\mu_0 > 0$, reduction factor $\tau > 1$, stopping tolerance $\epsilon > 0$.
Initialization: Set $x = x_0$, $\mu = \mu_0$. Ensure x_0 lies strictly in the interior of the feasible region ($h(x_0) > 0$).
while Stopping criteria are not met **do**
    Solve the following barrier problem using some linear solver:

$$\min_x \phi(x, \mu) = f(x) - \mu \sum_{i=1}^{m} \ln(h_i(x))$$

    Update the solution x to the result of the barrier problem.
    Reduce the barrier parameter:

$$\mu = \frac{\mu}{\tau}.$$

    Check for convergence:
    if The KKT residuals are below ϵ: **then**
        Stop. Return x as the solution.
    end if
end while

3.1.3 Modeling Network Constraints

While defining the optimization problem in its entirety, key components of the problem's constraints were modeled using neural-network-based approximations in order to maximize computational efficiency while making constraints differentiable. This comprised field prediction or, more specifically, spherical harmonic coefficient prediction $\boldsymbol{c}(\boldsymbol{I}(t))$ in Equation 3.3a, as well as smallest singular value prediction $\sigma_{\min}(\cdot)$ in Equation 3.3b. The benefit of differentiable constraints is that they enable the implementation of gradient descent algorithms for optimization, which provide a variety of advantages, particularly with regard to runtime and computational efficiency.

Spherical harmonic coefficient prediction is used for magnetic field estimation within the sphere radius $R = 4.5$ cm in this work. Spherical harmonics are a class of orthogonal functions defined on the surface of a sphere. These functions, denoted by $Y_l^m(\theta, \phi)$, where $l \in \mathbb{N}$ and $m \in \mathbb{Z}$ are integers representing the degree and order of the harmonic, respectively, and $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$ are the polar and azimuthal angles, respectively, naturally lend themselves to problems exhibiting spherical symmetry, such as the generation of magnetic fields in MPI scanners. The fields induced by running currents through the coils of the corresponding field generators can be expressed as a linear combination of these spherical harmonics, with the coefficients of the individual harmonics representing the strength of the respective harmonic in the field. When working with spherical harmonics, particularly with regard to computational efficiency, the concept of *t-designs* becomes a valuable asset. A set

of $N \in \mathbb{N}$ points defined on the surface of a sphere is called a t-design if the integral of any polynomial p of degree $k := \deg p \leq t,\ t \in \mathbb{N}$ over the sphere can be approximated[4] by a discrete sum over these points. In the context of spherical harmonic expansions, selecting a suitable t-design can significantly reduce computational overhead while preserving accuracy.

Field Prediction

Given the soft iron core's impact on magnetic field generation, development of a non-linear model for the current-to-field relationship was essential. A neural network is used to approximate this relationship, utilizing training data derived from empirical measurements of the fields generated under various currents. The model implemented as part of this thesis expands on findings from [11], which proposes a model architecture, data pre-processing, and a training strategy for magnetic field prediction. The model architecture used for training is presented in Figure 3.2. The implemented model was trained using 9150 adjusted training samples, optimized to minimize prediction error across different regions of the selection field generator's Field Of View (FOV) and thereby providing an accurate basis for field predictions. Training samples were generated through field simulation in the COMSOL Multiphysics Software [12]. As in the model implementation proposed in [11], both input (currents to the four top-left coils of the left coil grid) and output (spherical harmonic coefficients defining the corresponding magnetic field) data was normalized using z-score normalization. Training was performed for a total of 250 epochs, after which the model weights yielding the lowest loss during training were used for inference.

The scope of this model's applicability for field prediction is extended from modeling the magnetic field generated when applying currents to the four top-left coils of the selection field generator's left coil grid to modeling the magnetic field generated when also applying mirrored currents to the four top-left coils of its **right** coil grid, see Figure 3.3. This is achieved by augmenting the spherical harmonic expansion polynomials with a transformation about the y-axis (the axis along which the grids face each other). This involves negating the y-coordinate of the position in field space γ_y for each field direction, negating the y-components of the spherical harmonic coefficients c_y for the magnetic field in y-direction, and finally adding the resulting polynomials to the original polynomials. More formally, given the original spherical harmonic expansion polynomials

$$B_i = f(\boldsymbol{\gamma}, c_i) := f(\gamma_x, \gamma_y, \gamma_z, c_i),\ i \in \{x, y, z\}, \tag{3.9}$$

augmentation is performed, resulting in the following definitions:

$$B_j = f(\gamma_x, \gamma_y, \gamma_z, c_j) + f(\gamma_x, -\gamma_y, \gamma_z, c_j),\ j \in \{x, z\}, \tag{3.10a}$$

$$B_y = f(\gamma_x, \gamma_y, \gamma_z, c_y) + f(\gamma_x, -\gamma_y, \gamma_z, -c_y). \tag{3.10b}$$

[4] More specifically, the integral of p is equal to the average value of p over the set of N points.

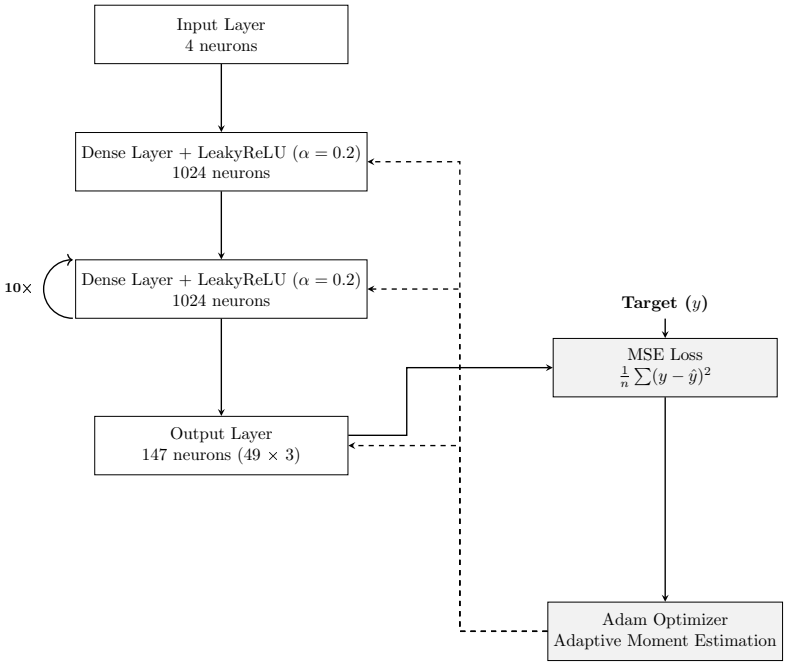


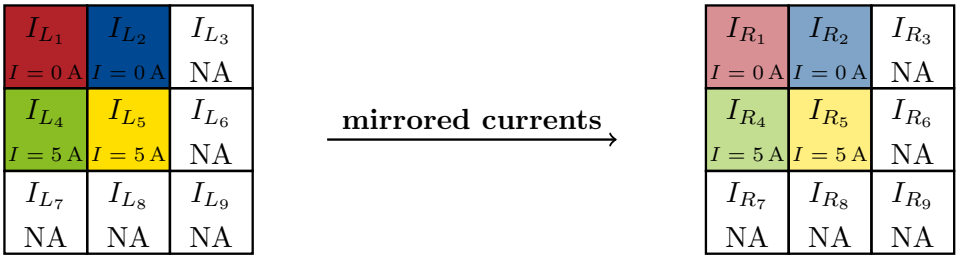
Figure 3.2: Visualization of model architecture, optimizer, and loss function for field prediction model

Negating the y-coordinate of the position in field space is a necessary augmentation to each of the spherical harmonic expansion polynomials, since $\gamma_y = 0$ defines the plane exactly in between and parallel to the coil grids of the selection field generator. To this end, negating the y-coordinate mirrors the position in field space about this plane, allowing the corresponding field prediction model output to mimic the strength of the magnetic field in the original position in field space that would be generated if only the four top-left coils in the right coil grid were active. Negating the y-components of the spherical harmonic coefficients is a further necessary augmentation specifically to the expansion polynomial for the y-direction, since the magnetic field generated by the right coil grid is exactly inverted to that generated by the left coil grid in the y-direction.

Using this method of augmentation, it is effectively possible to model a total of eight input currents in coils on both sides of the selection field generator with a field prediction model originally designed for four input currents to coils solely on the left coil grid of the field generator.

Smallest Singular Value Prediction

The Singular Value Decomposition (SVD) is known to be a computationally expensive procedure (time complexity of $O(mn \cdot \min(m,n))$ given a matrix $M \in \mathbb{R}^{m \times n}$). Although matrix dimensions are fixed and thus do not scale with any physical quantity of the system used in this work, the runtime associated with executing the Singular Value Decomposition using a



(a) Left coil grid (b) Right coil grid

Figure 3.3: Visualization of the coil grid setup with simulative mirroring of the currents of the left coil grid to the right coil grid

numerical approach is still substantial, since this operation is executed frequently as part of the optimization algorithm. To this end, an approach comprising a differentiable and lower runtime algorithm for the SVD using a neural network was proposed. More specifically, based on the constraint definition in Equation 3.3b, a neural network was trained for the prediction of the smallest singular value in a symmetric 3×3 matrix.

Training a model for the purpose of smallest singular value prediction first involved generating synthetic training data resembling practical measurements using Wigner's semicircle distribution for the generation of eigenvalues for random Hermitian matrices [13], as well as the Haar distribution for the generation of eigenvectors for random unitary matrices [14]. Wigner's semicircle distribution serves as an adequate distribution for random eigenvalues, since Wigner's semicircle law states that the distribution of eigenvalues in symmetric $n \times n$ matrices whose entries are independent and identically distributed random variables resembles the shape of a semicircle (particularly as n goes to infinity) [15], defined by the following probability density function:

$$f(x) = \begin{cases} \frac{2}{\pi R^2}\sqrt{R^2 - x^2} & \text{if } |x| \leq R, \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

The Haar distribution then serves as an adequate distribution for corresponding eigenvectors, since the Haar measure provides a uniform distribution over the group of orthogonal matrices [16]. Training data is ultimately generated by diagonalization as follows:

$$Q\Lambda Q^{-1}, \ \Lambda := \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \tag{3.12}$$

where Q is the matrix containing the Haar-distributed eigenvectors column-wise and $\lambda_1, \ldots, \lambda_n$ are the eigenvalues pulled from Wigner's semicircle distribution.

Random symmetric matrices are generated to provide testing data using a separate process to assess the robustness of this model, in this case initializing matrices via Gaussian orthogonal ensembles [17]. Gaussian orthogonal ensembles provide an adequate model for random

symmetric matrices, since they are described by the Gaussian measure on the space of square real symmetric matrices [18].

The difference in generated matrices between the training and testing data can be best visualized by plotting their respective eigenvalue distributions, which also illustrates the general magnitude of eigenvalues, see Figure 3.4.

Figure 3.4: Eigenvalue distributions of training and testing data for gradient estimation model

Model architectures and training parameters were defined based on findings from [19]. The model architecture, optimizer, and loss function used for training is presented in Figure 3.5.

Finally, training was performed using the six upper triangle matrix values as input and the smallest singular value of the matrix as output. Before training the final model, a preliminary model was trained using all three singular values as output to establish a proof of concept. Twelve plots visualizing the predictive accuracy of this preliminary model can be found in the Appendix under Figure 6.1. For the final model, training and validation loss curves encountered over the course of 5000 training epochs are presented in Figure 3.6. The curves indicate that, though improvements to model accuracy stagnate after approximately 512 epochs, *overfitting*, i.e., a reduction in model generalization capabilities due to fixation on the training data, is avoided throughout all 5000 epochs. After training, model accuracy was further assessed by evaluating the relative difference between the true smallest singular value and that predicted by the model across the aforementioned testing data. This difference is further compared to a baseline model for smallest singular value prediction that simply outputs the mean smallest singular value across the testing data regardless of the matrix values it receives as input, yielding the results presented in Table 3.1.

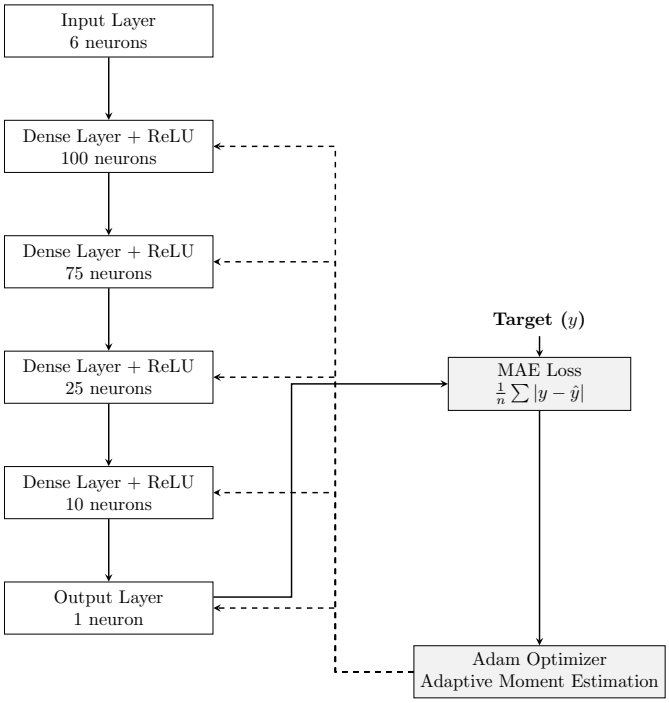


Figure 3.5: Visualization of model architecture, optimizer, and loss function for smallest singular value prediction model

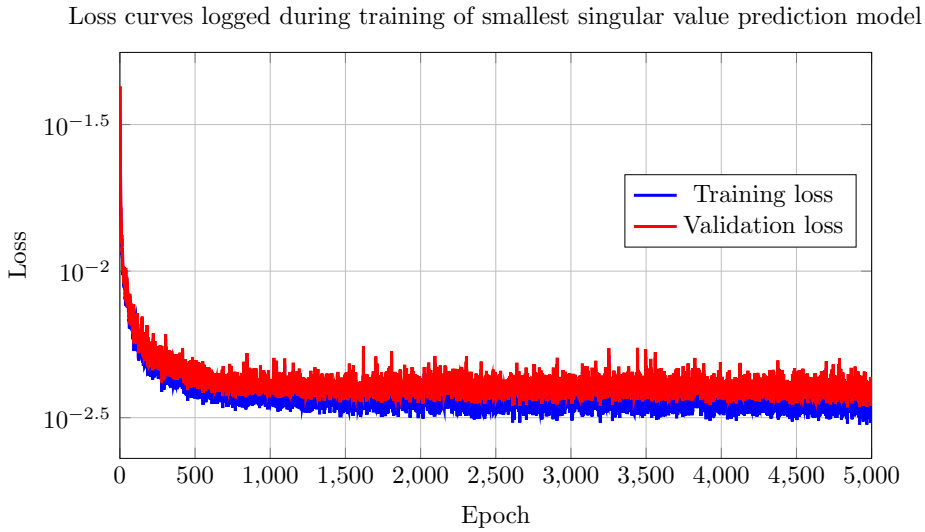


Figure 3.6: Loss curves for training of smallest singular value prediction

Difference metric	Trained model	Baseline model
Mean	0.0033	0.1095
Std. dev.	0.0029	0.0828
Max.	0.0457	0.7171

Table 3.1: Comparison of smallest singular value prediction accuracy between trained model and baseline model

3.2 Numerical Setup

The target FFP trajectory used for analysis is defined as follows:

$$\boldsymbol{\gamma}(t) := \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} \frac{-0.04}{9}\,(t-1)\,\mathrm{m} \\ 0 \\ 0 \end{bmatrix}, \; t \in [1, 10], \tag{3.13}$$

where t indexes the sequence of points along this trajectory by temporal discretization with time steps Δt. The magnitude of the time steps is not explicitly defined, since the only time-dependent aspect of the optimization problem, namely the constraint on the change in current, refers to the differences in current between each step in the corresponding sequence rather than between some two discrete points in time. The points of the trajectory are visualized in Figure 3.7.

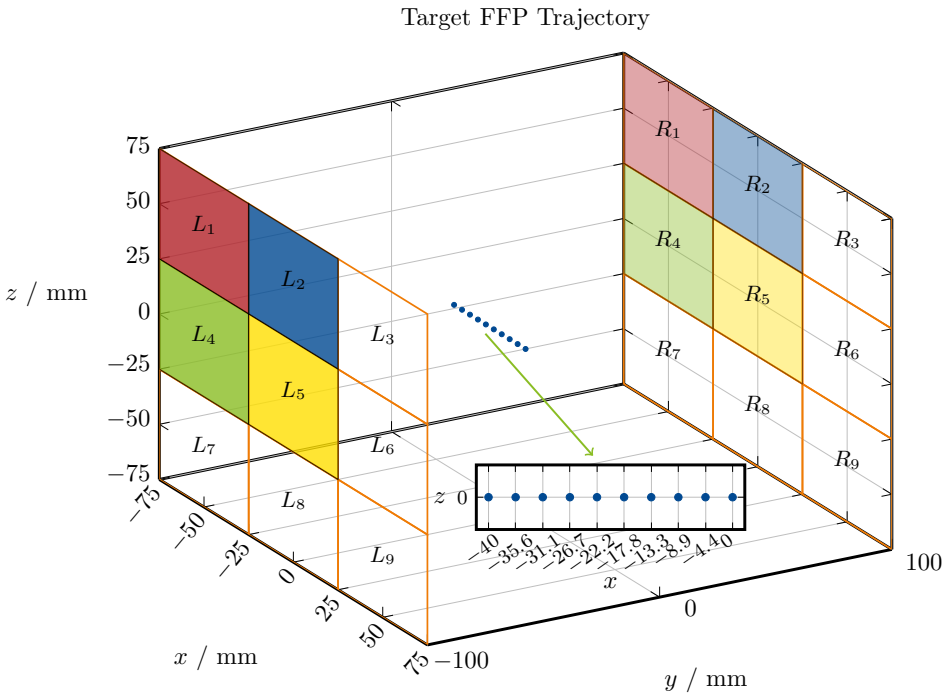


Figure 3.7: Visualization of points on target FFP trajectory

Initial constraints to the optimization problem are defined as follows:

$$0\,\mathrm{A} \leq I_n(t) \leq 10\,\mathrm{A},\ n \in [1,8], \tag{3.14}$$

$$\left|\dot{I}_n(t)\right| \leq 5\,\mathrm{A}/\Delta t,\ n \in [1,8], \tag{3.15}$$

$$\left|B_i\left(\boldsymbol{\gamma}\left(t\right), \boldsymbol{c}\left(\boldsymbol{I}\left(t\right)\right)\right)\right| \leq 2.5 \times 10^{-5}\,\mathrm{T},\ i \in \{x,y,z\}, \tag{3.16}$$

$$\sigma_{\min}\left(J_r B|_{(\boldsymbol{\gamma}(t),\boldsymbol{c}(\boldsymbol{I}(t)))}\right) \geq 0.1\,\mathrm{T/m}. \tag{3.17}$$

In an effort to improve numerical stability and balance the influence of each of these constraints, the constraint bounds are normalized to absolute values of one [20]. The better-conditioned constraint definitions are as follows:

$$-1\,\mathrm{A} \leq \frac{I_n(t) - 5}{5} \leq 1\,\mathrm{A},\ n \in [1,8], \tag{3.18}$$

$$-1\,\mathrm{A}/\Delta t \leq \frac{\dot{I}_n(t)}{5} \leq 1\,\mathrm{A}/\Delta t,\ n \in [1,8], \tag{3.19}$$

$$-1\,\mathrm{T} \leq B_i(\boldsymbol{\gamma}(t), \boldsymbol{c}(\boldsymbol{I}(t))) \cdot 2.5 \cdot 10^5 \leq 1\,\mathrm{T},\ i \in \{x,y,z\}, \tag{3.20}$$

$$\sigma_{\min}\left(J_r B|_{(\boldsymbol{\gamma}(t),\boldsymbol{c}(\boldsymbol{I}(t)))}\right) \cdot 10 \geq 1\,\mathrm{T}/m. \tag{3.21}$$

As a further improvement to the definition of the optimization problem, based on previous empirical trials in COMSOL, the currents are initialized as $I_{L_1}(t) = I_{L_2}(t) = I_{R_1}(t) = I_{R_2}(t) = 0\,\mathrm{A}$ and $I_{L_4}(t) = I_{L_5}(t) = I_{R_4}(t) = I_{R_5}(t) = 5\,\mathrm{A}$ for each time step $t \in [1, 10]$. This initialization, due to symmetry, leads to an FFP at approximately [−0.025 m 0 0], i.e., approximately at the center of the FFP trajectory. Visualizing the magnetic field based on spherical harmonic coefficients computed by the field prediction model verifies this assertion regarding the position of the FFP, see Figure 3.8.

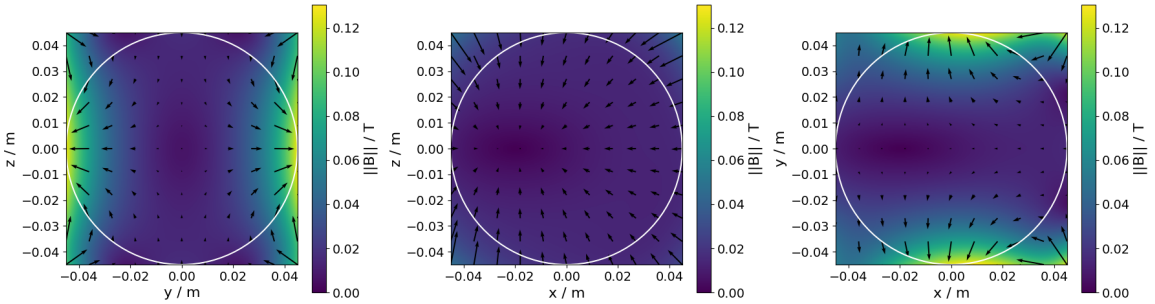


Figure 3.8: Visualization of predicted magnetic field

3.3 Implementation

The final implementation of the optimal control algorithm is based entirely on Julia [21], a programming language designed for numerical computing, particularly in scientific contexts. The key packages responsible for each of the crucial aspects of the optimal control algorithm are listed in Table 3.2.

Aspect of Implementation	Responsible Package(s)
Design & Solution of Optimization Problem	**JuMP** [22], **Ipopt** [5]
Neural Network Design, Training, & Inference	**Flux** [23] [24]
Magnetic Field Prediction	**SphericalHarmonicExpansions** [25], **DynamicPolynomials** [26]

Table 3.2: Statistical characteristics of tokenized sequences (auto-regressive)

Since the field prediction model was originally trained in Python [27] using PyTorch [28], a custom conversion script was executed to transfer the model architecture to Flux. The exact script can be found in the Appendix under Listing 6.1.

Using an implementation of the control algorithm in Julia, it is possible to optimize current sequences for given trajectories and constraint parameters on consumer-grade CPU hardware. The CPU used in this work is an AMD Ryzen 5 PRO 5675U.

3.4 Analysis

The results of the current sequence optimization for the numerical setup are first evaluated on the basis of the magnitude of the post-optimization value of the objective function (power consumption) and the accuracy of the field sequence generated by the optimized current sequence (FFP offset), since these metrics are most relevant for practical applications. Subsequently, these metrics are analyzed together with algorithmic efficiency (iteration count and computation time) and model accuracy (deviation in gradient prediction) for comparison across various degrees of gradient constraint (Equation 3.3b). As part of this analysis, particular focus is placed on the trade-off between reducing the FFP offset and minimizing power consumption.

4 Results

4.1 Numerical Setup

The optimization result for the numerical setup defined in Section 3.2 yielded a current sequence that provided a basis for the analysis of FFP offset, gradient prediction accuracy, and power consumption. FFP offset and stepwise power consumption are visualized in addition to the FFP trajectory produced by the optimized currents in Figure 4.1.

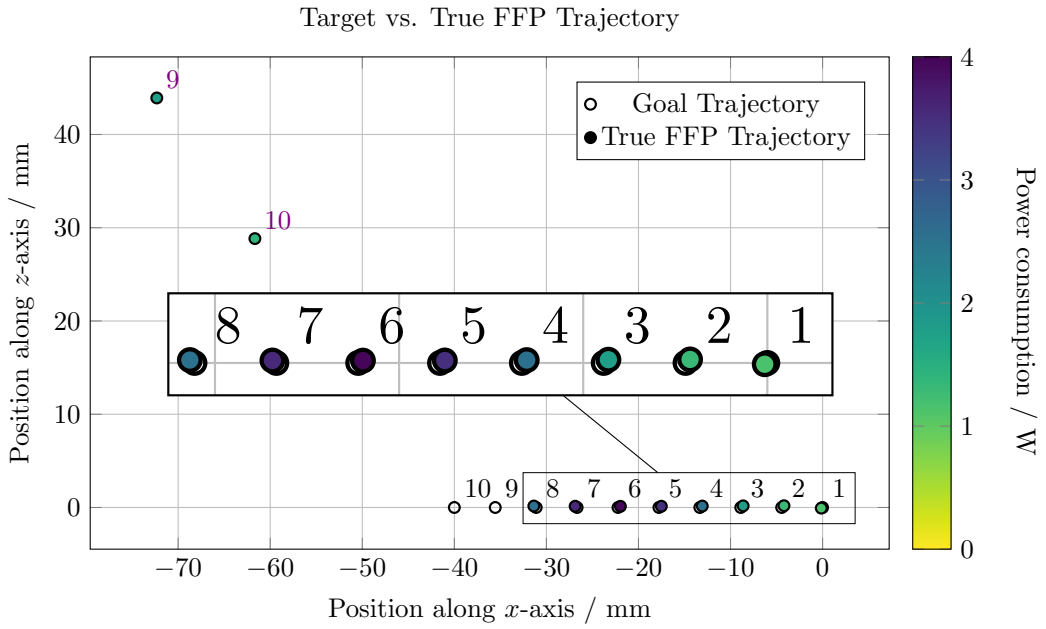


Figure 4.1: Visualization of target and result FFP trajectory

The plot in Figure 4.1 displays unusual behavior for the last two (ninth and tenth) steps of the FFP trajectory of the optimized current sequence. More specifically, these points on the trajectory, while maintaining low respective power consumption, produce FFP offsets of magnitudes that should not be achievable according to Equation 3.6. The explanation for this behavior lies in the fact that the final points of the target FFP trajectory approach the bounds of the sphere defined by the spherical harmonics (radius $R = 4.5 \times 10^{-3}$ m) too closely. To

account for this, a subsequent analysis was performed specifically for only the first **eight** sequence steps, visualized in Figure 4.2.

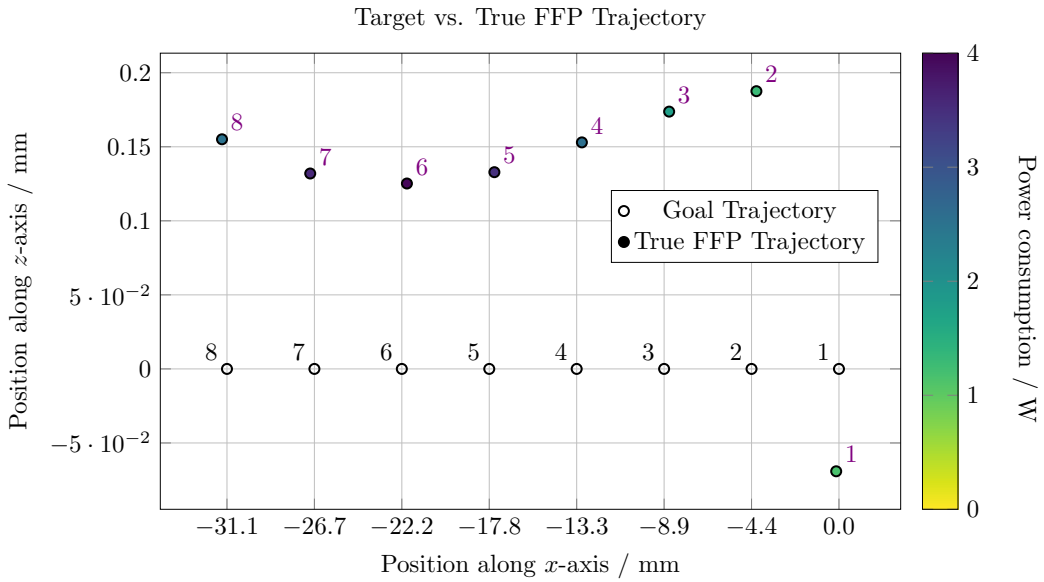


Figure 4.2: Visualization of target and result FFP trajectory (first 8 steps in sequence)

From the plot in Figure 4.2, it is clear that the FFP offset is much more uniform for the first eight steps of the trajectory, with magnitudes at or below the maximum expected offset according to Equation 3.6. Stepwise power consumption also behaves as expected, with peak consumption reached at the seventh point on the trajectory, i.e., at $x = 0.02\overline{6}$. This stands to reason because this is the closest point along the trajectory to being directly between the center and leftmost coils of the coil grid (recall that since the coils in the grid are spaced 5 cm apart center-to-center, the exact point along the x-axis is located at $x = 0.025$). Creating an FFP between these coil pairs requires an elevated level of power consumption due to the requirement on the fields to cancel each other out at a point which is farthest from any given coil center than is the case for all other points on the trajectory. For the points around this point on the trajectory, the power consumption falls monotonically and approximately symmetrically.

4.2 Current Sequence Optimization

A series of six optimizations with varying degrees of gradient constraint was performed for the same eight-step trajectory as detailed in the numerical setup in Section 4.1, with constraint definitions ranging from $g = 0.025\,\mathrm{T/m}$ to $g = 0.5\,\mathrm{T/m}$. Constraint values are uniformly logarithmically distributed (base two) from 0.025 to 0.4 T/m, with the additional value of 0.5 T/m serving to illustrate the limits of the trade-off between minimizing power consumption and minimizing FFP offset. In addition to the iteration count and the computation time, the

termination status obtained with IPOPT is listed for each optimization in Table 4.1. A list of all possible termination criteria with corresponding explanations can be found in the Appendix under Table 6.1.

g (in T/m)	Iter.	Time (in s)	Iter. per second	Termination status
0.025	263	224.488	1.172	Almost locally solved
0.05	166	138.044	1.203	Almost locally solved
0.1	111	75.447	1.471	Almost locally solved
0.2	253	253.867	0.997	Almost locally solved
0.4	281	274.579	1.023	Locally infeasible
0.5	90	58.159	1.759	Locally infeasible

Table 4.1: Overview of iteration count, computation time, and termination status for performed optimizations

The measurements in Table 4.1 illustrate that while the number of iterations performed during optimization fluctuates between the defined constraint values, the number of iterations performed per second is consistently around 1 for each of the optimizations performed given the utilized CPU.

In addition to optimization time and iteration count, power consumption, FFP offset, and gradient prediction accuracy were analyzed for each optimization. The corresponding values are listed in Table 4.2. Due to fixed current initialization defined in Section 3.2, these results are obtained in deterministic fashion. Gradient prediction accuracy is assessed by calculating the relative deviation of the predicted smallest singular value of the gradient from its true smallest singular value at each point on the trajectory, defined as follows:

$$\frac{\left|\sigma_{\min_{\text{pred}}} - \sigma_{\min_{\text{true}}}\right|}{g}, \tag{4.1}$$

where $\sigma_{\min_{\text{pred}}}$ and $\sigma_{\min_{\text{true}}}$ are the predicted and true smallest singular values and g is the gradient constraint.

g (in T/m)	Mean power consumption (in W)	Mean FFP offset (in mm)	Mean relative gradient deviation
0.025	11.63	0.903	0.16
0.05	0.56	0.629	0.073
0.1	2.33	0.276	0.042
0.2	17.67	0.14	0.03
0.4	92.54	0.016	0.029
0.5	127.53	0.125	0.025

Table 4.2: Overview of power consumption and FFP offset for performed optimizations

The most notable outcomes in terms of finding the optimal trade-off between power consumption and FFP offset can be identified near $g = 0.1\,\mathrm{T/m}$. This can be demonstrated by plotting the mean FFP offset against the mean power consumption for each of the optimization results, see Figure 4.3. The relative gradient deviation decreases monotonically as the gradient constraint increases, although these measures are not quite inversely proportional. This indicates that there is a certain degree of absolute error in addition to the expected relative error for the outputs of the smallest singular value prediction model.

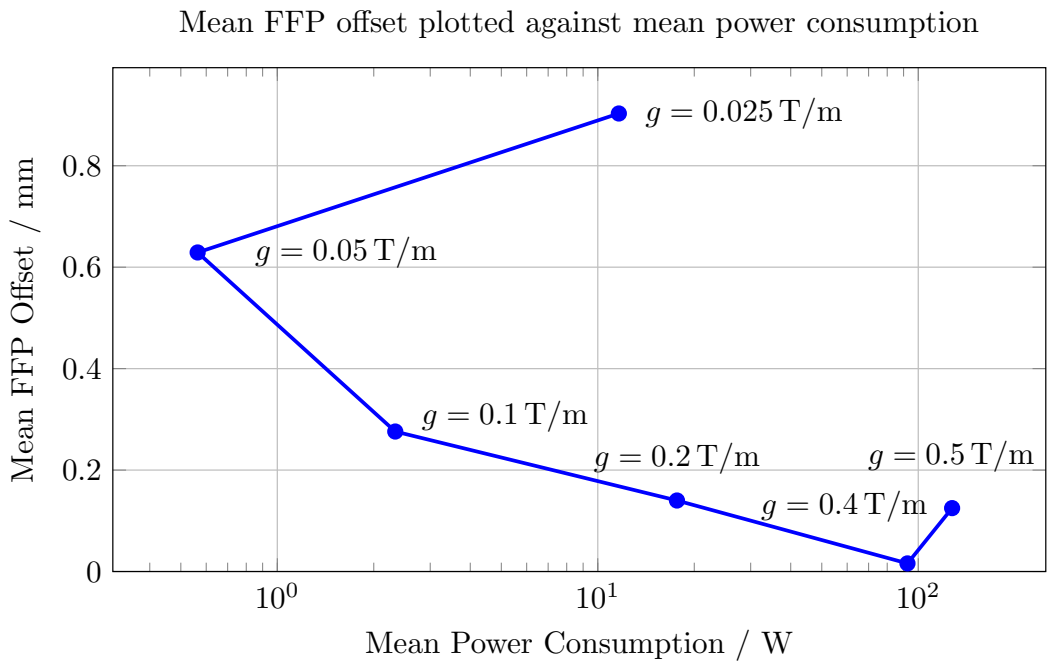


Figure 4.3: Visualization of the trade-off between mean FFP offset and mean power consumption for optimization with varying degrees of gradient constraint

Using the knee method on the plot in Figure 4.3, it can be reasoned that the optimization results for $g = 0.1\,\mathrm{T/m}$ and $g = 0.2\,\mathrm{T/m}$ provide desirable trade-offs between FFP offset and power consumption, assuming that these two measures are weighted similarly. Furthermore, the points for each extreme of the gradient constraint ($g = 0.025\,\mathrm{T/m}$ and $g = 0.5\,\mathrm{T/m}$) are each dominated by their nearest neighbor, thereby defining boundaries on the range of values for which the gradient constraint establishes a reasonable trade-off between FFP offset and power consumption. The optimization results for the constraint value $g = 0.5\,\mathrm{T/m}$ are poor because the strength of this constraint brings solving the optimization problem under this constraint to a point of near-infeasibility. On the other hand, the optimization results for the constraint value $g = 0.025\,\mathrm{T/m}$ are presumably poor due to a higher relative error in smallest singular value prediction (see the mean relative gradient deviation presented in Table 4.2), which could result from small diagonal matrix values associated with this low constraint value. Small diagonal matrix values for the gradient field are expected for a low constraint value, since the optimization algorithm attempts to find current sequences that

achieve minimal power consumption, which generally implies minimal gradient strength. Low matrix values may lead to a higher relative error in smallest singular value prediction in two ways. Firstly, the gradient estimation model is largely trained with different matrix values (the mean upper triangle matrix value in the training data is approximately 0.32) and may be unable to generalize for smaller values. Secondly, there may be some intrinsic absolute error in the model, leading to a higher relative error given smaller gradient values.

5 Discussion & Outlook

The experiments in this work provide a proof of concept that near-optimal current sequences for given field sequences can be found consistently in minutes or even seconds on consumer-grade CPU hardware. The experiments illustrate that current sequences with power consumption as low as —and even below —the single-digit Watt range can be determined by the optimization algorithm, which is a promising result. Moreover, arbitrarily low power consumption can be achieved within limits in exchange for loss in FFP precision. Conversely, arbitrarily hier FFP precision can be achieved within limits in exchange for higher power consumption.

The results achieved by the developed control algorithm are made possible mainly by the neural-network-based surrogate models for the forward problem, i.e., field prediction and gradient estimation. Not only are these models highly efficient[1], they are differentiable and thus suitable for optimization with gradient-based solvers. To this end, the use of a gradient-based solver presents an opportunity for further improvements regarding runtime and solution optimality.

For future work, the experimental setup could be extended from controlling four pairs of mirrored coils to controlling all 18 coils of the selection field generator independently. Furthermore, experimentation with different FFP trajectories could provide additional insight into practical ranges of constraint values for optimization. Finally, varying the bounds of other constraint definitions, such as the constraint for the magnetic field strength (Equation 3.3a), could provide further useful results and potentially yield improvements regarding power consumption and/or FFP offset.

[1]The field prediction model brings the execution time of approximately four minutes using COMSOL to an execution time on the order of a single millisecond.

Acknowledgments

I would like to express great appreciation to my advisers, Dr. Martin Möddel and Fynn Förger, for their support and encouragement. I would also like to thank Paul Jürß for his important contributions to this work, as well as the IBI team for their help in presenting this work at the 14th International Workshop on Magnetic Particle Imaging (IWMPI). Finally, I am grateful to the fantastic adviser of my previous research project, Dr. Jens Zemke, for his role as a second examiner for this thesis, and to Prof. Dr.-Ing. Tobias Knopp for making this thesis possible.

6
Appendix

Criterion	Description	Default Value
tol	Convergence tolerance for termination if NLP error is below threshold.	10^{-8}
s_max	Scaling threshold for NLP error.	100
max_iter	Maximum allowed iterations before termination.	3000
max_wall_time	Maximum wall-clock time allowed.	10^{20}
max_cpu_time	Maximum CPU time allowed.	10^{20}
dual_inf_tol	Absolute tolerance for dual infeasibility.	1
constr_viol_tol	Tolerance for constraint and bound violation.	0.0001
compl_inf_tol	Absolute tolerance for complementarity conditions.	0.0001
acceptable_tol	"Acceptable" convergence tolerance level.	10^{-6}
acceptable_iter	Iterations allowed before "acceptable" termination.	15
acceptable_dual_inf_tol	"Acceptable" threshold for dual infeasibility.	10^{10}
acceptable_constr_viol_tol	"Acceptable" constraint violation tolerance.	0.01
acceptable_compl_inf_tol	"Acceptable" complementarity condition tolerance.	0.01
acceptable_obj_change_tol	Objective function change stopping criterion.	10^{20}
diverging_iterates_tol	Threshold for detecting diverging iterates.	10^{20}
mu_target	Final value of barrier parameter for termination.	0

Table 6.1: Termination criteria for IPOPT

```julia
using Flux
using BSON
using NPZ

# Define the Flux.jl model architecture
function create_model(num_neurons::Int, num_layers::Int)
    layers = []
    for i in 1:num_layers
        push!(layers, Dense(num_neurons, num_neurons))
        push!(layers, x -> leakyrelu.(x, 0.2))
        push!(layers, Dropout(0.0))  # No dropout in your PyTorch model
    end

    return Chain(
        Dense(4, num_neurons),
        x -> leakyrelu.(x, 0.2),
        layers...,
        Dense(num_neurons, 49 * 3)
    )
end

# Initialize the Flux.jl model with the PyTorch model's weights and biases
function convert_model(model::Chain, WEIGHTS_DIR)
    for (i, layer) in enumerate(model.layers)
        # Check if weight and bias file exists
        if isfile(joinpath(WEIGHTS_DIR, "layer_$(i - 1)_weights.npy"))
            weights = npzread(joinpath(WEIGHTS_DIR, "layer_$(i - 1)_weights.npy"))
            bias = npzread(joinpath(WEIGHTS_DIR, "layer_$(i - 1)_bias.npy"))

            model_weights, model_bias = Flux.params(layer)
            model_weights .= weights
            model_bias .= bias
        end
    end
end

# Create the model
num_neurons = 2^10
num_layers = 10
model = create_model(num_neurons, num_layers)

WEIGHTS_DIR = joinpath(@__DIR__, "..", "..", "examples", "weights")

convert_model(model, WEIGHTS_DIR)

# Save the converted model
BSON.@save "model.bson" model
```

Listing 6.1: Converting PyTorch model to Flux.jl

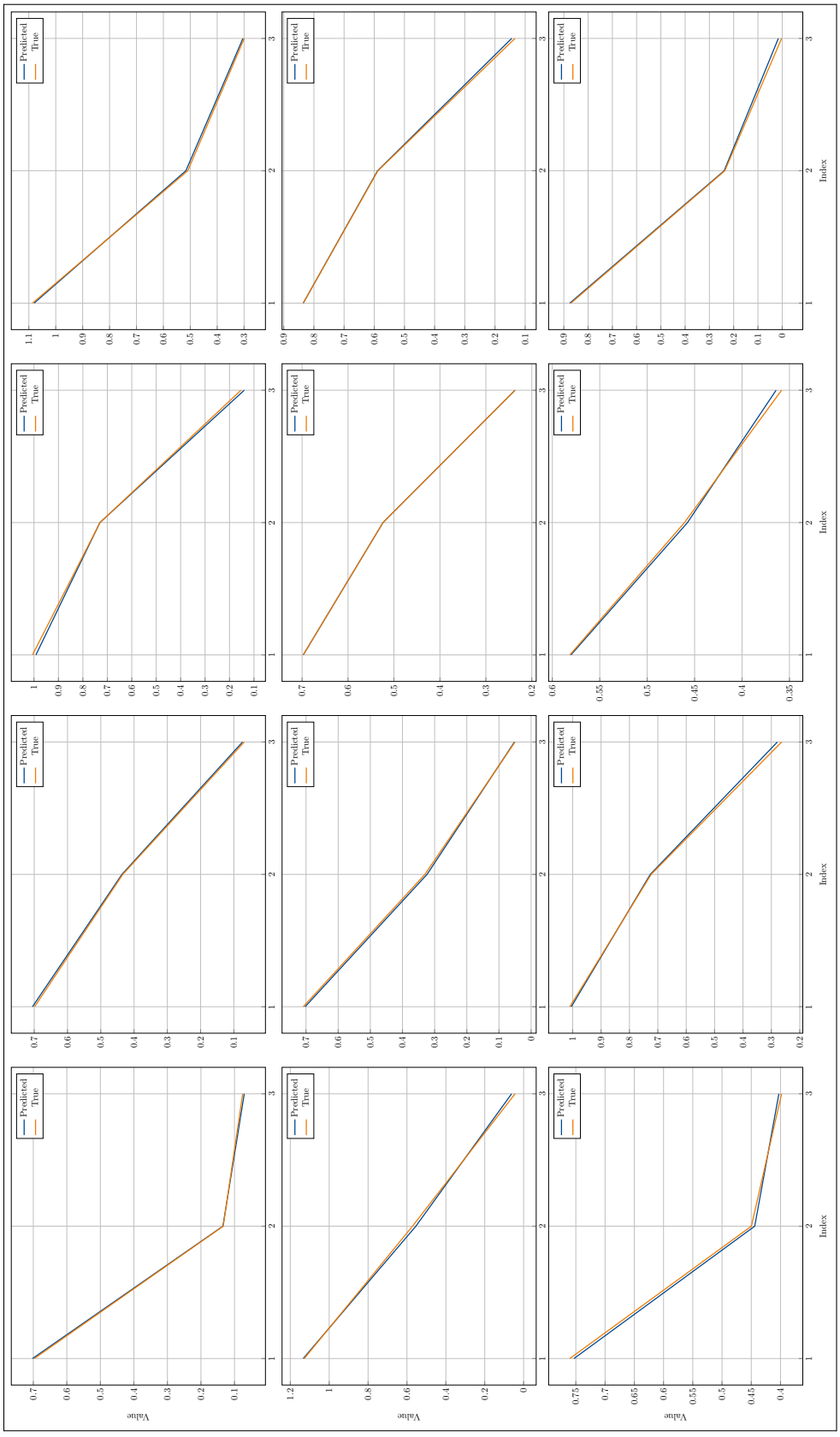


Figure 6.1: Example plots visualizing singular value prediction accuracy

Bibliography

[1] Fynn Foerger et al. “Low-power Iron selection and focus field generator”. In: *IJMPI* 8.1 Suppl 1 (2022).

[2] Tobias Knopp, Nadine Gdaniec, and Martin Möddel. “Magnetic particle imaging: from proof of principle to preclinical applications”. In: *Physics in Medicine & Biology* 62.14 (2017), R124.

[3] Marija Boberg, Tobias Knopp, and Martin Möddel. *Unique Compact Representation of Magnetic Fields using Truncated Solid Harmonic Expansions*. 2023. arXiv: 2302.07591 [physics.med-ph]. URL: https://arxiv.org/abs/2302.07591.

[4] Ying Xin et al. “Superconductors and Lenz’s law”. In: *Superconductor Science and Technology* 33.5 (2020), p. 055004.

[5] Andreas Wächter and Lorenz Biegler. “On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming”. In: *Mathematical programming* 106 (Mar. 2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[6] A. Wächter and Carnegie Mellon University. Department of Chemical Engineering. *An Interior Point Algorithm for Large-scale Nonlinear Optimization with Applications in Process Engineering*. Carnegie Mellon University, 2002. URL: https://books.google.dk/books?id=pLDW0AEACAAJ.

[7] H. W. Kuhn and A. W. Tucker. “Nonlinear programming”. In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. Berkeley and Los Angeles: University of California Press, 1951, pp. 481–492.

[8] P.R. Amestoy et al. “A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling”. In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41.

[9] P.R. Amestoy et al. “Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures”. In: *ACM Transactions on Mathematical Software* 45 (1 2019), 2:1–2:26.

[10] Joseph W. H. Liu. “The Multifrontal Method for Sparse Matrix Solution: Theory and Practice”. In: *SIAM Review* 34.1 (1992), pp. 82–109. ISSN: 00361445, 10957200. URL: http://www.jstor.org/stable/2132786 (visited on 01/25/2025).

[11] Fynn Foerger et al. “Current-to-Field Prediction for Non-Linear Magnetic Systems via Neural Networks”. In: *IJMPI* 11.1 Suppl 1 (2025). under review.

[12] COMSOL Multiphysics. “Introduction to COMSOL multiphysics®”. In: *COMSOL Multiphysics, Burlington, MA, accessed Feb* 9 (1998), p. 2018.

[13] L. Arnold. “On Wigner’s semicircle law for the eigenvalues of random matrices”. In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 19.3 (Sept. 1971), pp. 191–198. ISSN: 1432-2064. DOI: 10.1007/BF00534107. URL: https://doi.org/10.1007/BF00534107.

[14] K. Wieand. “Eigenvalue distributions of random unitary matrices”. In: *Probability Theory and Related Fields* 123.2 (June 2002), pp. 202–224. ISSN: 1432-2064. DOI: 10.1007/s004400100186. URL: https://doi.org/10.1007/s004400100186.

[15] Tianchong Jiang. “Wigner’s semicircle law for Gaussian random matrices”. In: *Chicago University* (2021).

[16] Magnus Lundberg and Lennart Svensson. “The Haar measure and the generation of random unitary matrices”. In: *Processing Workshop Proceedings, 2004 Sensor Array and Multichannel Signal*. IEEE. 2004, pp. 114–118.

[17] Jose Angel Sanchez Gomez and Victor Amaya Carvajal. *Uniqueness of the Gaussian Orthogonal Ensemble*. 2019. arXiv: 1901.09257 [math.PR]. URL: https://arxiv.org/abs/1901.09257.

[18] Vladimir Igorevich Bogachev. *Gaussian measures*. 62. American Mathematical Soc., 1998.

[19] Derek Xu et al. “SV-Learn: Learning Matrix Singular Values with Neural Networks”. In: *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2022, pp. 232–237. DOI: 10.1109/ICDMW58026.2022.00039.

[20] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006.

[21] Jeff Bezanson et al. “Julia: A fresh approach to numerical computing”. In: *SIAM review* 59.1 (2017), pp. 65–98.

[22] Miles Lubin et al. “JuMP 1.0: Recent improvements to a modeling language for mathematical optimization”. In: *Mathematical Programming Computation* 15 (2023), 581–589. DOI: 10.1007/s12532-023-00239-3.

[23] Michael Innes et al. “Fashionable Modelling with Flux”. In: *CoRR* abs/1811.01457 (2018). arXiv: 1811.01457. URL: https://arxiv.org/abs/1811.01457.

[24] Mike Innes. “Flux: Elegant Machine Learning with Julia”. In: *Journal of Open Source Software* (2018). DOI: 10.21105/joss.00602.

[25] *SphericalHarmonicExpansions.jl: A Julia package to handle spherical harmonic functions*. Version 0.1.3. URL: https://github.com/hofmannmartin/SphericalHarmonicExpansions.jl.

[26] Benoît Legat. “Multivariate polynomials in Julia”. In: *JuliaCon*. July 2022. URL: https://pretalx.com/juliacon-2022/talk/TRFSJY/.

[27] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[28] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.