universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Physics-Informed Extreme Learning Machines for efficient Poisson equation solutions

verfasst von | submitted by

Thomas Kroiss

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von | Supervisor:	Dipl.-Ing. Dr.techn. Lukas Exl Privatdoz.

Abstract

The use of smaller Neural Networks to solve the Poisson equation is a valuable alternative to classical approaches due to their speed and efficiency. Traditional numerical methods, such as the finite element method (FEM) and finite difference method (FDM), can suffer from high computational cost, ill-conditioning, and difficulties in handling higher dimensions. Machine learning-based approaches, particularly Physics-Informed Neural Networks (PINNs) and Extreme Learning Machines (ELMs), offer alternative strategies for solving Partial Differential Equations (PDEs) without requiring explicit mesh-based discretization. This thesis investigates the development of efficient solvers for the Poisson equation using Physics-Informed Extreme Learning Machines (PIELMs). PIELMs combine the rapid training capability of ELMs with physics-based constraints, enabling the efficient approximation of PDE solutions. A key focus of this thesis is the choice of basis functions for the PIELM architecture, which directly affects numerical stability, accuracy, and computational efficiency. The study systematically compares various basis functions, including Radial Basis Functions (RBFs), B-Splines, Chebyshev polynomials, and Fourier series, analyzing their impact on solution quality. Additionally, different sampling strategies—such as Poisson Disk Sampling, Sobol Sequences, and Uniform Sampling-are examined to determine their influence on the condition number of the basis function system matrix. The experimental results demonstrate potential advantages and drawbacks of each of those combinations. In addition, boosting techniques are explored as a means to enhance solution accuracy. The findings of this research contribute to the advancement of physics-informed machine learning methods, demonstrating their potential to provide fast, accurate, and scalable solutions for PDEs. Future research directions include adaptive basis function selection, extending PIELMs to high-dimensional and time-dependent PDEs and tackling the stray field problem in micromagnetism.

Kurzfassung

Die Verwendung kleinerer neuronaler Netze zur Lösung der Poisson-Gleichung ist aufgrund ihrer Geschwindigkeit und Effizienz eine gute Alternative zu klassischen Ansätzen. Traditionelle numerische Methoden, wie die Finite-Elemente-Methode (FEM) und die Finite-Differenzen-Methode (FDM), können unter hohen Rechenkosten, schlechter Konditionierung und Problemen beim Lösen von Gleichungen in höheren Dimensionen leiden. Auf maschinellem Lernen basierende Ansätze, insbesondere physikinformierte neuronale Netze (PINNs) und extrem lernende Maschinen (ELMs), bieten alternative Strategien zur Lösung partieller Differentialgleichungen (PDEs), ohne dass eine explizite Diskretisierung mittles Grids erforderlich ist. In dieser Arbeit wird die Entwicklung effizienter Löser für die Poisson-Gleichung mit Hilfe von physikalisch informierten extrem lernenden Maschinen (PIELMs) untersucht. PIELMs kombinieren die schnelle Trainingsfähigkeit von ELMs mit physikbasierten Einschränkungen und ermöglichen so eine effiziente Approximation von PDE-Lösungen. Ein Schwerpunkt dieser Arbeit ist die Untersuchung und Auswahl von Basisfunktionen für die PIELM-Architektur, die sich direkt auf die numerische Stabilität, Genauigkeit und Berechnungseffizienz auswirkt. Die Studie vergleicht systematisch verschiedene Basisfunktionen, einschließlich Radialer Basisfunktionen (RBFs), B-Splines, Tschebyscheff-Polynome und Fourier-Reihen, und analysiert deren Einfluss auf die Lösungsqualität. Darüber hinaus werden verschiedene Domänen-Sampling-Strategien - wie Poisson Disk Sampling, Sobol-Sequenzen und einheitliche Gitter - untersucht, um ihren Einfluss auf die Zustandszahl der Basisfunktions-Systemmatrix zu bestimmen. Die experimentellen Ergebnisse zeigen, dass Poisson Disk Sampling die numerische Stabilität in RBF-basierten PIELMs durch Verringerung des Anstiegs der Zustandszahlen deutlich verbessert, während Tschebyscheff-Polynome und B-Splines gut konditionierte Approximationen liefern, die für strukturierte und adaptive Diskretisierungen geeignet sind. Fourier-Reihen sind zwar in periodischen Bereichen wirksam, zeigen aber bei der Lösung allgemeiner PDEs aufgrund von Überanpassungseffekten bei hochfrequenten Komponenten Grenzen. Zusätzlich dazu werden Boosting-Techniken als Mittel zur Verbesserung der Lösungsgenauigkeit untersucht. Die Ergebnisse dieser Forschung tragen zur Weiterentwicklung von physikalisch informierten maschinellen Lernmethoden bei und zeigen deren Potenzial, schnelle, genaue und skalierbare Lösungen für PDEs zu liefern. Zukünftige Forschungsrichtungen umfassen adaptive Basisfunktionsauswahl, die Erweiterung von PIELMs auf hochdimensionale und zeitabhängige PDEs und die Lösung des Streufeldproblems im Mikromagnetismus.

Acknowledgment

I gratefully acknowledge financial support by the Austrian Science Foundation (FWF) through the projects of Doz. Dr. Exl at WPI and Univ. Wien: "Data-driven Reduced Order Approaches for Micromagnetism" (Data-ROAM) (Grant-DOI: 10.55776 / PAT7615923) at WPI, "Design of Nanocomposite Magnets by Machine Learning" (DeNaMML) (Grant-DOI: 10.55776 / P35413) at Univ. Wien research platform MMM "Mathematics – Magnetism – Materials".

I would also like to express my sincere gratitude to my supervisor, PD Dr. Lukas Exl, for his invaluable guidance and support throughout the development of this thesis. His expertise and insights have played a crucial role in shaping my research direction. I am also deeply grateful to Dr. Sebastian Schaffer from the MMM group at the University of Vienna, whose feedback and professional guidance significantly contributed to my understanding of the subject matter.

While this thesis presents an investigation into Physics-Informed Extreme Learning Machines (PIELMs) for solving the Poisson equation, there were inherent constraints that limited the scope of additional explorations. Time constraints imposed by my academic curriculum has made it difficult to conduct further investigations into boosting techniques, incremental learning strategies, alternative geometric configurations, and additional algorithmic approaches. Although these aspects offer promising directions for future research, the necessity to meet specific submission deadlines within my curriculum framework required a focused and efficient approach to the core research questions of this thesis. Despite these limitations, the findings presented establish a foundation for continued exploration of adaptive basis function selection, extended solution architectures, and enhanced learning methodologies in future studies. I remain hopeful that subsequent research will build upon these results and further advance the field of machine learning-based PDE solvers.

I also acknowledge the use of Open AI's Chat GPT for assistance with language editing and paraphrasing in the preparation of this manuscript.

Finally, I extend my appreciation to my family, friends and especially my girlfriend for their unwavering support and encouragement throughout the construction of the thesis.

Contents

1 Introduction and Motivation

The increasing relevance of computational problems in physics and engineering has driven the need for efficient and accurate numerical methods for solving Partial Differential Equations (PDEs). PDEs describe a wide range of physical phenomena, from heat conduction and fluid dynamics to electromagnetism and quantum mechanics [9, 12]. Among these, the Poisson equation plays a fundamental role in various applications, particularly in magnetostatics and micromagnetism, where it governs the relationship between magnetic potential and its sources [26].

Partial Differential Equations are fundamental in numerous areas of science and engineering, forming the mathematical basis for modeling complex physical phenomena. They also describe how quantities such as heat, electromagnetic fields, fluid flow, and potentials vary in space and time [9]. The Poisson equation, which describes how source terms affect resulting potential fields, crucial in various physical contexts such as electrostatics, fluid dynamics, gravitation, and particularly magnetostatics, is the focus of this thesis. In the context of magnetostatics and micromagnetics, the Poisson equation describes the magnetostatic potential generated by magnetic charges. Accurate and efficient solutions to the Poisson equation are indispensable for understanding and predicting the behavior of magnetic materials. Such insights are crucial in various technological applications including the development of magnetic sensors, data storage devices, and advanced magnetic structures.

Micromagnetism investigates the behavior of magnetic materials and magnetic structures on scales ranging from nanometers to micrometers. This field primarily seeks to understand magnetization states, magnetic domain structures, and magnetization dynamics. Central to micromagnetic modeling are PDE-based formulations, such as the Landau-Lifshitz-Gilbert (LLG) equation, which describes magnetization dynamics and energy minimization problems. Within these problems, the Poisson equation plays a crucial role in determining the scalar magnetostatic potential from the magnetic charge distribution. The magnetostatic potential obtained from the Poisson equation directly determines the magnetic field distributions, profoundly impacting the magnetic properties and dynamics of micromagnetic systems. Thus, efficient numerical solutions to this equation are not only essential for fundamental research but also have significant technological implications, particularly for the development of advanced magnetic sensors, and also the stray field problem in micromagnetism.

Traditionally, PDEs are solved using numerical techniques such as the finite element method (FEM) or finite difference method (FDM) [3, 10]. While these methods are well-established, they suffer from high computational cost and ill-conditioning, especially in high-dimensional domains or also in complex geometries, where very fine grids are demanded. Particularly in micromagnetism, where complex geometries and strongly varying material properties are present, conventional numerical methods often reach their limits. [44, 14]. The challenges of classical numerical methods lead to consider also different approaches in practice. The high computing times limit the scalability and thus often prevent extensive parameter studies and realistic, high-resolution simulations, which would regularly be necessary in micromagnetics [37]. At the same time, the poor condition numbers cause numerical solution methods to converge more slowly or become unstable, which requires additional numerical measures for stabilisation that further increase the already high computational effort [39]. This not

only reduces the reliability of the simulation results, but also makes the application to many realistic scenarios almost impossible or severely limited. These practical limitations of traditional numerical methods therefore directly motivate the research and development of new, innovative approaches.

With the rise of machine learning, new approaches for solving PDEs have emerged, offering data-driven and physics-informed alternatives [33, 27]. Neural networks, in particular, have gained attention for their ability to approximate complex functions and their potential to overcome some of the limitations of traditional numerical solvers [15, 17]. One promising class of machine learning approaches for solving PDEs are Physics-Informed Neural Networks (PINNs) [33, 17], which embed physical laws directly into the loss function of a neural network, by calculating the actual residual in respect to the PDE. However, PINNs rely on iterative optimization, making them computationally expensive and difficult to tune.

An alternative is the Extreme Learning Machine (ELM) [23, 22], a type of single-layer feed-forward neural network that trains much faster than traditional Neural Networks by avoiding iterative weight optimization. However, standard ELMs do not inherently enforce physical constraints, limiting their direct applicability to PDEs. To address the limitations of both PINNs and ELMs, Physics-Informed Extreme Learning Machines (PIELMs) have been proposed [8]. PIELMs integrate physical constraints directly into the network structure, ensuring that the solutions satisfy the governing equations. This approach eliminates the need for computationally costly penalty terms and enables fast, efficient training [8, 39].

This thesis explores the use of PIELMs for solving the Poisson equation, particularly in the context of micromagnetics. The goal is to assess how well PIELMs perform across different geometric domains, configurations, and boundary conditions. Beyond solving the Poisson equation, this work aims to set the groundwork of extending PIELMs to more complex applications, particularly in magnetostatics and micromagnetism. One promising direction is the integration of PIELMs with the splitting ansatz of Garcia-Cervera and Roma, which decomposes PDE problems into subproblems that are solved separately within the magnetic domain [37]. This technique, originally developed for whole-space transmission problems, can be leveraged to enhance the efficiency and stability of PIELM-based solvers, particularly in scenarios requiring continuity across multiple domains or interface conditions.

The aim of this thesis is therefore to investigate Physics-Informed Extreme Learning Machines (PIELMs) as an efficient and accurate alternative to solving the Poisson equation. A particular focus is on analysing different basis functions and sampling strategies, which are investigated with regard to their numerical stability and solution quality. The knowledge gained will be used to enable and optimise the use of PIELMs in micromagnetism and similar fields.

A key aspect of PIELMs is the optimal choice of basis functions in the context of the PDE problem, which directly targets the improvement of numerical stability, accuracy, and computational efficiency. This thesis systematically investigates the application of different basis functions—including Radial Basis Functions (RBFs), B-Splines, Chebyshev polynomials, and Fourier series—to determine their impact on PIELM-based Poisson solvers.

Beyond classical approached PIELMs, boosting techniques can also be leveraged to improve solution accuracy. Inspired by gradient boosting in machine learning, these techniques iteratively refine the approximation by adding corrective terms, improving accuracy without significantly increasing computational cost [16]. This work aims to combine the efficiency of ELMs, the physics-awareness of PINNs, and the accuracy-improving capabilities of other basis functions and boosting to develop fast and accurate PDE solvers for potential real-world applications in micromagnetics.

Motivated by these insights, the primary objective of this thesis is to systematically evaluate and optimize Physics-Informed Extreme Learning Machines (PIELMs) for efficient and accurate solutions of the Poisson equation within the context of micromagnetism. Specifically, the thesis addresses the following research aims:

- Identify which basis functions (e.g., Radial Basis Functions, B-splines, Chebyshev polynomials, Fourier series) are most suitable for improving the conditioning and accuracy of PIELMs when solving the Poisson equation?
- Can the implementation and evaluation of the new PIELM approaches in combination with different geometric shapes and domains provide significant improvements?
- To what extent can boosting techniques (Gradient Boosting) enhance the accuracy and computational efficiency of Physics-Informed Extreme Learning Machines?

These questions will be thoroughly addressed through experimental analyses across diverse geometric configurations to derive recommendations for the practical application of PIELMs within micromagnetics.

This thesis is structured as follows:

- **Chapter 2** provides a mathematical background, explaining PDEs, the Poisson equation, and traditional numerical methods. It also introduces machine learning approaches for PDEs, including PINNs and ELMs, and compares their advantages and drawbacks. Furthermore, PIELMs are introduced.
- **Chapter 3** presents hard constraint PIELMs as an efficient method for enforcing boundary conditions with the use of R-functions. In addition, RBFs and methods to declare RBF centers are explained.
- **Chapter 4** presents numerical results, evaluating the potentials of different basis functions and solution strategies.
- **Chapter 5** explores boosting techniques and their impact on PIELM performance.
- **Chapter 6** concludes the thesis and points to potential future research directions.

Although the focus of this thesis is on PIELMs, approaches such as Convex Incremental ELMs (CI-ELM) [21], Deep Operator Networks (DeepONets) [31] or Fourier Neural Operators (FNOs) [30] for solving PDEs are also interesting approaches. Due to limited time, these methods could not be investigated, but offer interesting perspectives for future research.

2 Neural Networks for PDEs

The following chapter first presents the theoretical background to Partial Differential Equations (PDEs), traditional solution methods and the machine learning methods used. This provides the basis for understanding the PIELM methods analyzed this thesis.

PDEs describe a wide range of physical processes, such as heat conduction, wave propagation, fluid dynamics, and electromagnetism. Unlike ordinary differential equations (ODEs), which depend on a single independent variable, PDEs involve functions of multiple variables and their partial derivatives. Because they govern fundamental physical laws, PDEs are used extensively in scientific and engineering applications [9, 12].

Although PDEs can sometimes be solved analytically, most real-world applications require numerical approximation techniques. However, these methods face computational challenges, including high memory requirements, instability, and ill-conditioning, which can make solving PDEs difficult in practice.

A general PDE takes the form:

$$F\left(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots\right) = 0. \tag{2.1}$$

PDEs are categorized based on their structure:

- **Order:** The highest derivative present determines whether the PDE is first-order, second-order, or higher [43].
- **Linearity:** A PDE is linear if it can be expressed as $L(u) = f(x)$, where L is a linear differential operator. If the equation contains nonlinear terms in u or its derivatives, it is classified as nonlinear [9].
- **Type:** Linear PDEs are divided into three main types:
  - *Elliptic PDEs:* Model equilibrium states and steady-state solutions. such as Poisson's equation.
  - *Parabolic PDEs:* Describe diffusive processes, such as the heat equation.
  - *Hyperbolic PDEs:* Govern wave propagation, such as the wave equation.

2.1 The Poisson Equation

One of the most important elliptic PDEs is the Poisson equation, which models how a quantity $u(x)$ is influenced by a given source term $f(x)$:

$$-\Delta u(x) = f(x), \quad x \in \Omega. \tag{2.2}$$

Where Δu is the Laplacian operator. The Poisson equation is used in electrostatics, gravitational potential theory, fluid mechanics, and micromagnetics [26, 28].

Although the Poisson equation has an analytical solution in simple cases, real-world numerical applications introduce several difficulties, for example:

- **High Computational Cost:** Solving in higher dimensions or discretizing the domain into a large number of elements increases the size of the system to be solved, leading to higher computational expense.
- **Ill-Conditioning:** The numerical system arising from discretization often has a large condition number, making iterative solvers slow and unstable.

2.2 Conditioning

An important aspect in solving PDEs and especially the Poisson equation numerically is *ill-conditioning*. This issue arises when small errors in input data or numerical round-off accumulate and significantly affect the final solution.

The condition number of a normal matrix A is defined as:

$$\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \tag{2.3}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest singular values of A. A high condition number means that small perturbations in the input can cause large deviations in the output [44].

For the Poisson equation, ill-conditioning arises for example due to grid refinement, because increasing the resolution (more grid points) makes $\kappa(A)$ larger, worsening stability.

These issues lead to several challenges in numerical solvers. Iterative methods such as Conjugate Gradient (CG) tend to converge more slowly when dealing with ill-conditioned matrices, as small eigenvalues hinder the stability of the algorithm. As a result, higher computational cost arises since more iterations are required to achieve convergence. In many practical applications, these problems are difficult to avoid, particularly when dealing with higher dimensions. This motivates the exploration of alternative approaches, such as machine learning-based methods, which can provide more robust and scalable solutions [25].

2.3 Traditional Numerical Methods and Limitations

Traditionally, Partial Differential Equations (PDEs) such as the Poisson equation are solved numerically using classical methods like the finite element method (FEM) or finite difference method (FDM). These methods have been extensively used in scientific and engineering applications due to their robustness and well-established theoretical foundations [29].

The finite element method divides the domain into smaller, simpler subdomains known as elements. Within each element, the solution is approximated by polynomial functions, typically linear or quadratic. The coefficients of these polynomials are determined by enforcing continuity conditions across element boundaries, resulting in a system of linear equations. FEM is particularly effective for problems involving complex geometries or boundary conditions because of its flexibility in mesh adaptation and local refinement capabilities [45].

The finite difference method discretizes the PDE directly by approximating derivatives at discrete points arranged in a structured grid. FDM replaces continuous derivatives by algebraic differences, which simplifies implementation and makes it computationally efficient for regular, structured domains. However, in irregular or curved domains, computational cost can increase because of the need to divide the domain in a high amount of small parts. [29, 36].

While providing strengths, traditional numerical methods also encounter challenges:

- **Computational Costs**: Potentially high computational resources required for very fine discretizations or complex geometries.
- **High-Dimensions**: Computational cost grows with dimensions, making grid-based methods challenging to apply beyond 3D or 4D due to memory and time requirements.

These challenges are also relevant in fields like micromagnetism, which requires precise modeling at very small scales and involves complex material interfaces. Specifically, the Poisson equation in micromagnetics is used for calculating magnetic potentials arising from given magnetization distributions. Accurate and efficient solutions to this PDE influence the applicability of simulations used to develop applications in micromagnetism [13].

The complexity of micromagnetic problems could lead to potentially larger computational cost when solved using traditional numerical methods in higher dimensions. The resulting linear systems are potentially characterized by larger condition numbers, where other methods could offer a slightly better approach to the problem.

Due to these challenges inherent in classical methods, there is a motivation for exploring alternative computational strategies, particularly machine learning-based approaches. Methods such as Physics-Informed Neural Networks (PINNs) and Extreme Learning Machines (ELMs) are particularly useful. They can be directly trained to a specific problem with governing equations and boundary conditions without relying on explicit discretization schemes, thereby potentially providing greater flexibility and improved numerical performance in some complex applications [33, 24].

This has prompted research towards innovative machine learning methods tailored for complex PDE problems, including those arising in micromagnetic simulations.

2.4 Advantages of Machine Learning for PDE Solving

Machine learning offers an alternative approach to solving PDEs. Neural networks provide a mesh-free approximation, directly learning from provided functions or data without requiring predefined grids. Once trained, these models enable fast evaluation of approximated solutions and offer the flexibility to adapt the architecture or training process to improve results.

Traditional solvers compute $u(x)$ for a given $f(x)$ each time a new problem is encountered. Specific machine learning methods, such as DeepONets [31], aim to learn a *solution operator*, mapping source terms $f(x)$ to solutions $u(x)$:

$$\mathcal{G} : f(x) \mapsto u(x). \tag{2.4}$$

This allows generalization, meaning a trained Neural Network can solve new PDEs efficiently without recomputing the entire numerical scheme. While classical numerical methods like FEM and FDM remain dominant, machine learning techniques such as PINNs and ELMs offer useful alternatives for efficient PDE solutions.

Machine learning approaches provide a complementary set of tools that can generally handle high-dimensional and computationally expensive scenarios. [5]. The next chapter explores how machine learning can enhance the ability to solve PDEs, with a focus on methods, efficiency, accuracy, and adaptability. For the general potential of machine learning being discussed, a deeper exploration of how Neural Networks specifically address the challenges of solving PDEs is essential.

2.5 Neural Networks Intro

In the previous section, we established the role of machine learning as an alternative to traditional numerical solvers for PDEs.

Neural networks have emerged as a promising tool for solving PDEs due to their universal approximation capabilities and ability to handle complex, high-dimensional functions. [18]. Neural networks can approximate the underlying function space directly, enabling mesh-free solutions that are adaptable to irregular geometries and discontinuous coefficients. This makes them particularly attractive for problems where traditional solvers face high computational cost or require large memory resources [33, 17].

A neural network is a computational model inspired by the structure and function of the human brain, consisting of interconnected layers of artificial neurons. These networks process input data through layers of nodes, where each node applies a mathematical operation, transforming the input into a feature representation. The primary components of a neural network include:

- **Input layer:** Accepts raw data, such as numerical values or features, and passes it to the network.
- **Hidden layers:** Perform nonlinear transformations using weights, biases, and an activation function, capturing complex patterns and relationships in the data.
- **Output layer:** Produces the final prediction or result, such as classification labels or continuous values.

Each neuron computes a weighted sum of its inputs and applies an activation function to determine its output:

$$z = \sum_{i=1}^{n} w_i x_i + b, \quad a = \sigma(z), \tag{2.5}$$

where w_i are the weights, x_i are the inputs, b is the bias term, z is the weighted sum, and $\sigma(z)$ is the activation function.

The **activation function** introduces nonlinearity into the network. Without it, the network would only be capable of representing linear transformations, regardless of its depth. Common activation functions include:

- **Sigmoid:** $\sigma(z) = \frac{1}{1+e^{-z}}$, maps z to a range between 0 and 1. While useful for probabilistic outputs, it suffers from the vanishing gradient problem.
- **ReLU (Rectified Linear Unit):** $\sigma(z) = \max(0, z)$, retains positive values and outputs zero for negative values. It is computationally efficient and addresses the vanishing gradient problem.
- **Tanh:** $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, maps z to a range between -1 and 1. It is symmetric around zero and is effective when negative values are meaningful.

By combining these nonlinear transformations across layers, Neural Networks can approximate highly complex relationships and also providing the possibility of universal function approximation [18, 15]. Neural networks are capable of learning complex mappings from input to output spaces. They approximate the unknown solution $u(x)$ by minimizing a loss function. A standard feedforward neural network consists of multiple layers, where each transformation is defined as:

$$u_\theta(x) = W_n \sigma(W_{n-1} \sigma(\dots \sigma(W_1 x + b_1) + b_2) \dots) + b_n. \tag{2.6}$$

where:

- x represents the input features (e.g., spatial coordinates).
- W_i and b_i are the trainable weights and biases of layer i.
- σ is a nonlinear activation function (e.g., ReLU, tanh).
- $u_\theta(x)$ is the neural network's output approximation of the PDE solution.

Through optimization, the parameters $\theta = \{W, b\}$ are adjusted to minimize the error between the predicted and actual solution.

A key advantage of machine learning and interesting research branch in PDE solving is that specific types Neural Networks, like DeepONets [31] are able to learn the operator $\mathcal{G}$, enabling them to generalize across different PDE instances and provide rapid solutions.

2.6 Physics-Informed Neural Networks (PINNs)

Physics-Informed Neural Networks (PINNs) represent a powerful framework for PDEs by embedding the governing physical laws directly into the training process of a neural network. This approach enables the solution of forward and inverse problems across domains, even in scenarios with sparse data or complex geometries. PINNs are particularly valuable for problems where traditional numerical methods struggle, such as solving PDEs in higher dimensions [33, 27].

Two well applicable examples of machine learning-based PDE solvers are:

- **Physics-Informed Neural Networks (PINNs)**, which enforce the governing equations via a physics-informed loss function.
- **Extreme Learning Machines (ELMs)**, which avoid iterative optimization by solving for weights in a single step.

Both of these methods have advantages and limitations, which motivates the development of Physics-Informed Extreme Learning Machines (PIELMs), a hybrid approach that combines the efficiency of ELMs with the physics-awareness of PINNs.

2.6.1 Solving PDEs with PINNs

In the context of PDEs, Physics-Informed Neural Networks (PINNs) approximate the solution $u(x)$ to a PDE defined on a domain Ω. Consider the Poisson equation as a representative example:

$$-\Delta u(x) = f(x), \quad x \in \Omega, \tag{2.7}$$

with prescribed boundary conditions $u(x) = g(x)$ on $\partial\Omega$. Here, $u(x)$ might represent the potential in a magnetostatic problem, where $f(x)$ is related to the source term.

The neural network $u_\theta(x)$, parameterized by weights and biases θ, is trained to approximate $u(x)$. The PDE and its boundary conditions are encoded as constraints within the network's loss function:

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}}, \tag{2.8}$$

where:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| -\Delta u_\theta(x_i) - f(x_i) \right|^2, \quad \mathcal{L}_{\text{BC}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| u_\theta(x_i) - g(x_i) \right|^2. \tag{2.9}$$

The total loss penalizes deviations from the governing equation and the boundary conditions. By minimizing this loss function, the network learns a function $u_\theta(x)$ that satisfies the physical constraints and provides a continuous solution across Ω [33, 27]. Here:

- The first term ensures the PDE residual is minimized.
- The second term enforces boundary conditions $\mathcal{B}u_\theta(x)$.

PINNs also leverage **automatic differentiation** [15], a technique that computes derivatives efficiently, avoiding finite difference approximations.

2.6.2 Advantages and Limitations of PINNs

Physics-Informed Neural Networks (PINNs) offer advantages in particular problem settings. By enforcing the governing equations within the loss function, PINNs incorporate boundary conditions, implementing physically laws. Even in areas with little or no data, they are able to produce precise predictions. They can approximate PDE solutions without requiring explicit domain discretization, making them flexible for complex geometries. Additionally, they scale well to high-dimensional problems, reducing the curse of dimensionality, which can be a problem in grid-based methods.

However, PINNs also face challenges. Training is computationally expensive, as it relies on iterative optimization, often requiring fine-tuning of hyperparameters such as learning rates and network depth. Their performance is sensitive to network architecture, making convergence potentially difficult for certain problems. These limitations motivate the exploration of alternative approaches like Extreme Learning Machines (ELMs), which aim to bypass iterative training while maintaining efficiency and accuracy.

2.7 Extreme Learning Machines (ELMs)

Extreme Learning Machines (ELMs) offer an alternative approach that avoids iterative training by solving for weights analytically. The core idea is to initialize random weights in the hidden layers and solve for the output weights in a single step [23]. ELMs are a type of single-hidden-layer feedforward neural network (SLFN) that provides efficient and scalable solutions to supervised learning problems. Introduced by [23], ELMs eliminate the need for iterative gradient-based optimization by freezing the weights and biases of the hidden layer. This approach significantly reduces training time while maintaining the universal approximation capability of Neural Networks [18] [22].

For an input $x \in \mathbf{R}^d$, an ELM with M hidden nodes and a scalar output is given by:

$$u_\beta(x) = \sum_{i=1}^{M} \beta_i \sigma_i(x) = \beta^T z(x). \tag{2.10}$$

where:

- M is the number of hidden neurons,
- $\sigma_i(x)$ is the activation function for the i-th hidden neuron.
- β is the output weight vector.
- $z(x)$ is the feature mapping in the hidden layer, defined as:

$$z(x) = [\sigma_1(x), \sigma_2(x), ..., \sigma_M(x)]^T. \tag{2.11}$$

The key innovation in ELMs lies in the optimization of the output weights β, which is performed using a closed-form solution based on the Moore-Penrose pseudoinverse. Although original ELMs use randomly generated feature spaces for approximation, their accuracy can be significantly enhanced through suitable feature transformations. Unlike traditional Neural

Networks, ELMs do not train through gradient descent. Employing multiple hidden layers results in non-convex optimization problems, leading to poor conditioning and difficulties in finding solutions. Instead, the optimal weights are computed using a least squares approach:

$$H\beta = Y, \tag{2.12}$$

where:

- H is the *hidden layer output matrix*, with elements $H_{ij} = \sigma_j(x_i)$.
- Y is the vector of target function values.
- The output weights β are solved via the Moore-Penrose pseudoinverse:

$$\beta = H^{\dagger}Y, \quad H^{\dagger} = (H^T H)^{-1} H^T. \tag{2.13}$$

To prevent ill-conditioning, Ridge regularization can be applied using Singular Value Decomposition (SVD):

$$H_{\mu}^{\dagger} = V S_{\mu}^{-1} U^T, \tag{2.14}$$

where S_{μ}^{-1} is a regularized inverse with singular values σ_i modified as:

$$S_{\mu,ii}^{-1} = \frac{\sigma_i}{\sigma_i^2 + \mu}. \tag{2.15}$$

The choice of basis functions in ELMs strongly influences the quality of the approximation. Different types of basis functions can be used to construct $\sigma_i(x)$:

- **Polynomial Basis Functions:**
$$\sigma_i(x) = x^i. \tag{2.16}$$
- **Chebyshev Polynomials:**
$$\sigma_i(x) = T_i(x), \quad \text{where } T_i(x) = \cos(i \cos^{-1}(x)). \tag{2.17}$$
- **Trigonometric Functions (Fourier Series):**
$$\sigma_i(x) = \sin(i\pi x), \quad \sigma_i(x) = \cos(i\pi x). \tag{2.18}$$
- **Radial Basis Functions (RBFs):**
$$\sigma_i(x) = \exp(-\gamma \|x - w_i\|^2). \tag{2.19}$$
- **B-Spline Basis Functions:**
$$\sigma_i(x) = B_i^k(x), \tag{2.20}$$
where $B_i^k(x)$ are B-spline basis functions of degree k.
- **Hyperbolic Tangent (Tanh):**
$$\sigma_i(x) = \tanh(x_i). \tag{2.21}$$

2.7.1 Advantages and Limitations of ELMs

Extreme Learning Machines (ELMs) offer advantages over traditional Neural Networks, particularly in terms of training speed. Unlike PINNs, which require iterative optimization, ELMs compute network weights in a single step, making them more efficient. Additionally, they avoid issues with hyperparameter tuning, reducing the complexity of network design. However, their accuracy strongly depends on the choice of activation functions or basis functions, which directly influence the quality of the approximation—an aspect explored in later Chapter 4.

Despite these benefits, ELMs have notable limitations. They lack built-in physics constraints, meaning solutions may not always satisfy the underlying PDEs unless additional corrections are applied. Furthermore, achieving robustness requires a careful selection of basis functions, as poor choices can lead to unstable or inaccurate solutions.

These challenges motivate the development of the more advanced Physics-Informed Extreme Learning Machines (PIELMs), which integrate physical constraints of PINNs while preserving ELMs' computational efficiency.

2.8 Comparison of PINNs vs. ELMs

ELMs differ from Physics-Informed Neural Networks (PINNs) in their computational approach, making them advantageous for PDE-based applications, particularly in large-scale systems. The key distinction lies in how parameters are optimized and physical constraints. PINNs rely on iterative training, where the network minimizes a composite loss function balancing data constraints and physics residuals. This process requires more iterations, making training computationally expensive and sensitive to hyperparameter selection. ELMs contrast, do not require iterative updates. Instead, after the hidden layer parameters (such as RBF centers or basis functions) are fixed, direct solving offers faster training speed. In contrast to PINNs, ELMs result in convex optimization problems, which bring advantages for discovering solutions. To highlight the differences between PINNs and ELMs, we summarize their key properties in Table 2.1.

Feature	**PINNs**	**ELMs**
Training Speed	Slower (iterative optimization)	Fast (closed-form)
Physics Constraints	Enforced via loss function	Not enforced by default
Optimization Problem	Non-convex	Convex (single hidden layer)
Convergence Stability	Sensitive to hyperparameters	Generally more stable

Table 2.1: Comparison between PINNs and ELMs.

Given these differences, the question arises as to how the two approaches can be combined in order to utilize the advantages of both methods and create a new on better method to solve PDEs. This leads to Physics-Informed Extreme Learning Machines (PIELMs).

PIELMs combine:

- The efficiency of ELMs for fast training.
- The physics constraints of PINNs for accurate solutions.

2.9 Physics-Informed Extreme Learning Machines (PIELMs)

Unlike conventional Neural Networks that require iterative backpropagation, ELMs utilize a *single hidden layer* with randomly initialized weights and solve for the output weights β directly via a least squares formulation.

Physics-Informed Extreme Learning Machines (PIELMs) extend the standard Extreme Learning Machine (ELM) framework by incorporating physical constraints while maintaining computational efficiency. Unlike standard Extreme Learning Machines (ELMs), where basis functions are trained purely on data without any underlying constraints, Physics-Informed Extreme Learning Machines (PIELMs) directly embed the governing physical laws into the training process. Instead of relying on iterative loss optimization, as seen in Physics-Informed Neural Networks (PINNs), PIELMs construct the solution space such that it inherently satisfies the underlying PDE. The core idea of Physics-Informed Extreme Learning Machines (PIELMs) is to combine the advantages of the rapid training capability of ELMs with explicit physical information derived directly from the PDE being solved.

In PIELMs, the learned function $u_\beta(x)$ must satisfy the **Poisson equation**:

$$-\Delta u_\beta(x) \approx f(x), \quad x \in \Omega. \tag{2.22}$$

Unlike conventional ELMs, which learn function approximations purely from data, PIELMs directly embed the governing PDE into the system, ensuring that the learned solution satisfies the underlying physical constraints. A key property of ELMs is that any linear operator applied to an ELM function is only applied to the activation functions, while the weights remain unchanged. This means that applying the Laplace operator to the ELM function:

$$u_\beta(x) = \sum_{i=1}^{M} \beta_i \sigma_i(x) \tag{2.23}$$

results in:

$$\Delta u_\beta(x) = \sum_{i=1}^{M} \beta_i \Delta \sigma_i(x). \tag{2.24}$$

This transformation is critical because it allows the Poisson equation to be directly incorporated into the system formulation without modifying the structure of the learned coefficients β. Instead of solving a purely data-driven function approximation, PIELMs enforce that the expansion satisfies the governing equation at a set of collocation points.

The corresponding system matrix Q is constructed as:

$$Q_{ij} = -\Delta \sigma_j(x_i), \tag{2.25}$$

leading to the modified Physics-Informed Least Squares system:

$$Q\beta = Y. \tag{2.26}$$

This ensures that the learned function satisfies the Poisson equation constraints, rather than being optimized based purely on data. Enforcing of boundary conditions is explained in Section 2.9.1 and in particular in Chapter 3 with the hard constraint method.

2.9.1 Quadratic Programming

A Physics-Informed ELM solution to the Poisson equation can be found by minimizing the corresponding energy functional:

$$u^*(x) = \arg\min_{u \in H_0^1(\Omega)} \frac{1}{2} \int_\Omega \Big(\|\nabla u(x)\|^2 - f(x)u(x) \Big)\, dx,$$

using a suitable set of activation functions.

In the PIELM setting, the approximate solution is represented as a linear combination of M basis (activation) functions $\sigma_j(x)$:

$$u_\beta(x) = \sum_{j=1}^{M} \beta_j \sigma_j(x), \tag{2.27}$$

where the PDE constraints are directly enforced by minimizing residuals in a physics-informed least squares sense. This leads to the quadratic optimization problem with linear constraints:

$$\min_{\beta \in \mathbf{R}^M} \frac{1}{2} \beta^T Q \beta - b^T \beta, \quad \text{subject to} \quad K\beta = z, \tag{2.28}$$

where matrices and vectors are defined explicitly by integrals as follows:

- **System matrix** Q, representing interactions among basis functions:

$$Q_{ij} = \int_\Omega \nabla \sigma_i(x) \cdot \nabla \sigma_j(x)\, dx \tag{2.29}$$

- **Source vector** b, representing the PDE's source term:

$$b_i = \int_\Omega f(x)\sigma_i(x)\, dx \tag{2.30}$$

- **Boundary constraint matrix** K **and vector** z, ensuring exact compliance with boundary conditions:

$$K_{ij} = \sigma_j(y_i), \quad z_i = g(y_i), \tag{2.31}$$

  evaluated explicitly at the boundary points $\{y_i\}_{i=1}^{N_{\partial\Omega}} \subseteq \partial\Omega$.

Thus, the Physics-Informed Extreme Learning Machine explicitly incorporates the PDE constraints and boundary conditions directly into its optimization framework, ensuring physically consistent and computationally efficient solutions [37].

While PIELMs provide an efficient and physics-aware framework for solving PDEs, their success depends also on correctly handling boundary conditions and domain constraints. By embedding boundary conditions directly into the function formulation, the hard constraint approach avoids the need for additional constraints within the linear system. The next chapter introduces hard constraint ELMs, explaining their mathematical foundation and how they enable robust boundary condition enforcement in ELMs and PIELMs.

3 Hard Constraint PIELMs

Solving PDEs and handling boundary conditions on complex domains is a significant challenge in numerical methods and physics-Informed machine learning. A strong ELM approach that can also be extended to PIELMs is the hard constraint formulation. The homogenous boundary conditions are enforced in the ELM's or PIELM's architecture. As a result, in many cases, the optimization issue is unconstrained and simpler to solve. First, a method to handle these boundary conditions is presented and will be used to further present the hard constraint method which is also used for the experimental parts in Chapters 4 and 5. A strong framework for representing complicated domains is provided by Constructive Solid Geometry (CSG). By applying set operations to basic geometric primitives like spheres, cubes, or their mixtures, CSG offers a methodical way to design complex shapes. It is feasible to effectively create intricate geometries that properly match the specifications of PDE-based simulations by utilizing CSG techniques [37].

3.1 R-Functions

R-Functions are real-valued and have the characteristic that their signs are entirely determined by the signs of their arguments, regardless of the magnitude of their arguments [41]. These functions enable to satisfy boundary conditions on all boundary points [35]. Introduced by [40], they provide a mathematical framework for defining geometric shapes using implicit functions. Instead of representing a shape by a set of discrete points, R-Functions describe the geometry through a continuous implicit function:

$$\ell(x) = \begin{cases} 0, & \text{if } x \in \partial\Omega, \\ > 0, & \text{if } x \in \Omega, \\ < 0, & \text{if } x \notin \Omega. \end{cases}$$

This approach enables a continuous representation of the geometry and allows for enforcement of homogenous Dirichlet boundary conditions.

These functions can be thought of as a type of Boolean algebra, in which true is denoted by a positive sign and false by a negative sign. These functions can be further built using the logical predicates, and since set theory and Boolean logic are related, geometric objects can be defined in this manner [37].

An example of a complete system of R-functions is described as followed:

- $\ell_1 \equiv -\ell_1$ **(Logical Negation)**
- $\ell_1 \wedge \ell_2 \equiv \min(\ell_1, \ell_2)$ **(Conjunction: Logical AND)**
- $\ell_1 \vee \ell_2 \equiv \max(\ell_1, \ell_2)$ **(Disjunction: Logical OR)**

To be able to form a complete system, all that is required is to specify conjunction and disjunction, hence logical negation is most often defined by real negation. From this follows

for conjunction:

$$\min(\ell_1, \ell_2) = \frac{1}{2}\left(\ell_1 + \ell_2 - \sqrt{(\ell_1 - \ell_2)^2}\right) \tag{3.1}$$

and similarly for disjunction:

$$\max(\ell_1, \ell_2) = \frac{1}{2}\left(\ell_1 + \ell_2 + \sqrt{(\ell_1 - \ell_2)^2}\right) \tag{3.2}$$

In theory, it is quite simple to represent the majority of geometric objects using implicit functions, but, as is the case with physics-informed machine learning, differential features are frequently crucial for modeling. The terms $\sqrt{(\ell_1 - \ell_2)^2}$ highlight that the operations show not-differentiability at the points where $\ell_1 = \ell_2$ [37]. Since we are concerned about differentiability, we want to solve this issue. To overcome this, new systems like R_0 and R_α have been introduced :

$$R_\alpha : \frac{1}{1+\alpha}\left(\ell_1 + \ell_2 \pm \sqrt{\ell_1^2 + \ell_2^2 - 2\alpha\ell_1\ell_2}\right), \quad -1 < \alpha \leq 1 \tag{3.3}$$

The R_0 system is particularly favored due to its differentiable nature, defined as:

$$R_0 : \ell_1 + \ell_2 \pm \sqrt{\ell_1^2 + \ell_2^2} \tag{3.4}$$

Here, + corresponds to disjunction (OR) and − corresponds to conjunction (AND). These forms ensure differentiability except at the point $\ell_1 = \ell_2 = 0$ [37].

3.1.1 Approximate Distance Function (ADF)

Since we are concerned about differentiability, especially in the context of Automatic Differentiation, it is critical to select a system of R-functions that satisfies differentiability throughout the domain Ω. The R_0 system meets this requirement. Differentiating the R_0 system with respect to its arguments ℓ_1 and ℓ_2 yields:

$$\partial\ell_i r = 1 \pm \frac{\ell_i}{\sqrt{\ell_1^2 + \ell_2^2}}, \quad i = 1, 2 \tag{3.5}$$

This expression is well-defined everywhere except at the singular point $\ell_1 = \ell_2 = 0$. On the boundary, where either $\ell_1 = 0$ or $\ell_2 = 0$, this derivative simplifies to well-defined values, ensuring differentiability at these points. A function ℓ is said to be normalized to the kth order if it satisfies:

$$\ell|_{\partial\Omega} = 0, \quad D_\nu\ell|_{\partial\Omega} = 1, \quad D_\nu^k\ell|_{\partial\Omega} = 0 \quad \text{for} \quad k = 2, \ldots, k, \tag{3.6}$$

where $D_\nu\ell$ denotes the directional derivative in the inward normal direction ν. Consequently, the gradient $\nabla\ell$ matches the inward normal vector ν, a beneficial property for normal vector computation. If a function ℓ is normalized at least to the first order, it closely approximates the exact distance function near the boundary. Such a function is termed an approximate distance function (ADF) [37].

Incorporating R-functions into PIELMs involves explicitly embedding geometric constraints within the computational model:

Solving PDEs within a domain Ω implicitly defined by an R-function $\ell(x)$ uses an ansatz form:

$$u(x) = \ell(x)\tilde{u}(x), \quad \ell(x) = 0 \quad \text{on} \quad \partial\Omega \tag{3.7}$$

For the solution structure of the Poisson problem 2.2, it is sufficient to use an ADF ℓ that is normalized to first order in our situation, where $r = 1$ [37].

3.2 Hard Constraint Method

Another approach and concrete realization is the **hard constraint approach** applied to the Poisson equation with Dirichlet boundary conditions. In the hard-constraint ELM setting, the approximate solution is explicitly formulated as:

$$u_\beta(x) = \ell(x)\hat{u}_\beta(x) + \tilde{g}(x) = \beta^T \left[\ell(x)z(x)\right] + \tilde{g}(x), \tag{3.8}$$

where:

- $\ell(x)$ is an indicator function that vanishes on the boundary $\partial\Omega$, i.e., $\ell(x) = 0$ for $x \in \partial\Omega$.
- $\tilde{g}(x)$ explicitly fulfills the boundary conditions, meaning $\tilde{g}(x) = g(x)$ on $\partial\Omega$.
- $z(x)$ represents the vector of chosen activation or basis functions $\sigma_j(x)$.

To explicitly embed the PDE into the training, the physics-informed loss function based on the PDE residual is minimized:

$$L_{\text{ELM}}(\beta) = \int_\Omega \left|\beta^T \Delta(\ell(x)z(x)) + \Delta\tilde{g}(x) + f(x)\right|^2 dx. \tag{3.9}$$

In practice, this integral is approximated using empirical risk minimization over a finite set of collocation points $\{x_i\}_{i=1}^N \subseteq \Omega$:

$$L_{\text{ELM}}(\beta) = \frac{1}{N}\sum_{i=1}^{N} \left|\beta^T \Delta(\ell(x_i)z(x_i)) + \Delta\tilde{g}(x_i) + f(x_i)\right|^2. \tag{3.10}$$

Minimizing this loss function leads to the linear system:

$$A\beta = b, \tag{3.11}$$

where the entries of matrix A and the vector b are defined explicitly by:

$$A_{ij} = -\Delta(\ell(x_i)\sigma_j(x_i)), \quad b_i = f(x_i) + \Delta\tilde{g}(x_i), \tag{3.12}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, M$. This formulation inherently ensures that boundary conditions are satisfied exactly and provides a computationally efficient and robust approach for solving PDEs in micromagnetic applications [37].

3.3 Radial Basis Functions (RBFs)

As described in Section 2.7, several different basis functions can be used to approximate solutions with ELMs or PIELMs. One strong choice of basis functions, which is described for example in [19] are Radial Basis Functions (RBFs). RBFs provide a flexible, mesh-free

approach to function approximation and numerical PDE solving. RBFs do not require explicit domain discretization, making them well-suited for PIELMs [6].

Radial Basis Functions (RBFs) depend only on the distance between the input point $\mathbf{x}$ and a center point $\mathbf{w}_j$. Mathematically, the RBF is expressed as:

$$\phi_j(\mathbf{x}) = \exp\left(-\gamma\|\mathbf{x} - \mathbf{w}_j\|^2\right), \tag{3.13}$$

where:

- $\mathbf{w}_j$ is the center of the RBF,
- $\|\mathbf{x} - \mathbf{w}_j\|$ is the Euclidean distance between the input point $\mathbf{x}$ and the center $\mathbf{w}_j$,
- $\gamma > 0$ controls the width of the RBF [6].

Key Properties:

- *Localization:* RBFs are highly localized, meaning their influence diminishes rapidly as the distance from the center increases.
- *Smoothness:* Gaussian RBFs are infinitely differentiable, making them ideal for representing solutions to PDEs that require smooth approximations.
- *Flexibility:* The parameter γ allows control over the level of locality. A larger γ results in tighter RBFs, capturing finer details, while a smaller γ creates broader RBFs, providing smoother approximations.

RBFs are widely used in interpolation, regression, and numerical PDE solving due to their ability to approximate smooth functions while remaining flexible in complex geometries.

3.3.1 Common Types of RBFs

Several types of RBFs exist, each with unique properties [6]:

- **Gaussian RBF:**
$$\phi(r) = e^{-\gamma r^2}. \tag{3.14}$$
Smooth and infinitely differentiable but sensitive to the parameter γ.
- **Multiquadric (MQ):**
$$\phi(r) = \sqrt{r^2 + \varepsilon^2}. \tag{3.15}$$
Provides good interpolation but can suffer from poor conditioning.
- **Inverse Multiquadric (IMQ):**
$$\phi(r) = \frac{1}{\sqrt{r^2 + \varepsilon^2}}. \tag{3.16}$$
Opposed to other methods, a higher shape parameter value leads to a less smooth surface.
- **Polyharmonic Spline (PHS):**
$$\phi(r) = r^k, \quad k \text{ odd}. \tag{3.17}$$
Does not require a shape parameter, making it more robust for scattered data interpolation.

3.3.2 Solution Operator for PDE Solving

A key advantage of RBFs is that they can approximate solution operators, enabling a general mapping from source terms to solutions [5]. Given a PDE of the form:

$$\mathcal{L}u(x) = f(x), \tag{3.18}$$

the goal is to learn an operator $\mathcal{G}$ such that:

$$u(x) \approx \mathcal{G}(f)(x) = \sum_{i=1}^{N} w_i \phi_i(x). \tag{3.19}$$

3.4 Sampling Methods for RBF Centers

Several strategies exist for selecting RBF centers. The accuracy of RBF-based methods strongly depends on how the RBF centers are distributed within the domain. Conventionally, the RBF nodes' centers are selected by random sampling. Selecting the best locations for the centers of the RBF nodes is more difficult than figuring out the width value $\gamma$to employ in an RBF neural network. The locations selected for the centers of an RBF network's nodes have a significant impact on how well the network can approximate a function. This is because each RBF node's effect is restricted to a particular area of the input space, and the size of this area of influence is determined by the width parameter of the node. The influence of a given RBF node on the network's output is strongest when the input vector is close to the node's center, and it decreases rapidly as the distance between them grows [42]. Therefore, poorly chosen centers can lead to worse approximations or potentially ill-conditioned systems, increasing numerical instability and error. Thus, selecting an appropriate sampling method is crucial in ensuring stability and accuracy. Different sampling methods, including quasi-random methods, will show different values for condition numbers and solution approximation, which is further investigated in Chapter 4.

3.4.1 Uniform Sampling

Uniform sampling is one of the simplest and most intuitive methods for creating RBF centers. Uniform sampling assigns an equal probability to every potential center point at each sampling instance. To generate random points using uniform sampling, random values are created for a point. Unlike quasi-random or structured sampling methods, uniform sampling only relies on the statistical properties of randomness for covering the domain.

The simplicity of uniform sampling makes it a choice for initial testing and comparative analysis of models. However, its lack of structural constraints can lead to uneven coverage of the domain, with clusters of points in some regions and sparse coverage in others. Therefore, while computationally efficient, uniform sampling can lead to a less accurate approximation and also potentially to a higher condition number. This properties are analyzed further in Section 4.5. In Figure 3.1 we can observe an example of points sampled with Uniform Sampling.

3.4.2 Sobol Sequence Sampling

Another approach to sample points, particularly useful when total coverage of the domain space is of advantage, is through the use of Sobol sequences [2] —a type of quasi-random

sequence. Sobol sequences, as described in [2], aim to minimize a property known as *discrepancy*, which quantitatively measures how evenly points are spread across a given multi-dimensional region.

As described in detail in [2], We aim to generate a set of well-distributed points $x^1, x^2, \ldots, x^N$ in a s-dimensional unit cube $[0,1]^s$. One way to measure the quality of distribution is discrepancy, which quantifies how uniformly points cover the space.

For a set of points $x^1, x^2, \ldots, x^N$ in $[0,1]^s$, and a rectangular region

$$G_{\mathbf{x}} = [0, x_1) \times [0, x_2) \times \cdots \times [0, x_s) \tag{3.20}$$

with volume $x_1 x_2 \ldots x_s$, the discrepancy is given by:

$$D^*(x^1, x^2, \ldots, x^N) = \sup |S_N(G_{\mathbf{x}}) - N x_1 x_2 \ldots x_s| \tag{3.21}$$

where $S_N(G_x)$ is the number of sampled points falling within the region G_x. The goal of Sobol sequences is to produce sequences of points with the smallest possible discrepancy. It is believed that Sobol sequences can not achieve a lower discrepancy than $\log^m n$.

For sampling RBF centers, Sobol Sequence sampling is particularly advantageous, because it systematically fills the input argument space and ensures that RBF centers are placed in a manner that better represents the entire space. This potentially improves approximation accuracy and computational efficiency compared to Uniform Sampling approaches. In Figure 3.1 we can observe an example of points sampled with Sobol Sequence.

3.4.3 Additive Recurrence Sequence

An Additive Recurrence Sequence (ARS), described in [34], is a method for generating sequences that provide low-discrepancy characteristics and sample a domain more completely, offering an alternative to traditional quasi-random sequences. It is particularly useful when more complete coverage of a domain is desired. By default, ARS does not require huge computational resources and covers the domain well, which makes it particularly attractive for scenarios involving high-dimensional spaces.

In one dimension, an additive recurrence sequence is defined as:

$$t_n = s_0 + n\alpha \pmod 1, \quad n = 1, 2, 3, \ldots \tag{3.22}$$

where:

- t_n represents the n-th term in the sequence.
- s_0 is the initial shift or starting point, often set to zero.
- α is an irrational number.

The choice of α significantly influences the uniformity of the sequence. Selecting α as the reciprocal of the golden ratio ϕ, defined as:

$$\phi = \frac{\sqrt{5}+1}{2} \approx 1.6180339887\ldots \tag{3.23}$$

minimizes discrepancy, leading to a more uniformly distributed sequence.

For sampling points in a multi-dimensional space, the additive recurrence method extends by assigning a distinct irrational number α_i to each dimension. The n-th point in a d-dimensional space is given by:

$$t_{n,i} = n\alpha_i \pmod 1, \quad \text{for } i = 1, 2, \ldots, d \tag{3.24}$$

where:

- t_n is the n-th point in the d-dimensional sequence.
- $s_{0,i}$ is the initial shift for the i-th dimension, typically set to zero.
- α_i is an irrational number associated with the i-th dimension.

To achieve low discrepancy in higher dimensions, it's beneficial to choose α_i based on the generalized golden ratio. Specifically, α_i can be selected as the fractional part of $1/\phi_{d,i}$, where $\phi_{d,i}$ is the unique positive root of the polynomial:

$$x^{d+1} = x + 1 \tag{3.25}$$

The simple implementation of ARS, combined with its avoidance of clustering and dense regions, considers it to be particular useful. It ensures that points t_n are distributed with low-discrepancy, making the sequence powerful for numerical integration. In Figure 3.2 we can observe an example of points sampled with ARS.

3.4.4 Poisson Disk Sampling

Poisson Disk Sampling, described in [4], is an adaptive sampling technique that ensures points are well-distributed with a minimum separation distance. Unlike purely random sampling, Poisson Disk Sampling creates quasi-random distributions and enforces:

- A minimum distance constraint, preventing points from being too close.
- A random yet uniform distribution, ensuring even coverage.

The algorithm works iteratively: 1. Start with an initial random point in the domain. 2. Generate candidate points in an annular region around existing points. 3. Accept a new point only if it is at least $d_{\min}$ away from all existing points. 4. Repeat until the domain is filled.

Let Ω be a computational domain in $\mathbf{R}^d$, and let $\mathcal{P}$ be a set of sampled points.

Poisson Disk Sampling is typically implemented using a bridging algorithm, such as Bridson's algorithm [4], which efficiently generates well-spaced points:

1. **Initialize the domain:** Define the computational region Ω.
2. **Select an initial random point** x_0 and place it in a list of active points.
3. **Generate candidate points:** For each active point x_{active}, sample k new candidate points in an annular region:

$$x_{\text{new}} = x_{\text{active}} + r(\cos\theta, \sin\theta), \tag{3.26}$$

   where:
   - r is randomly sampled from $[d_{\min}, 2d_{\min}]$.
   - θ is a random angle in $[0, 2\pi]$.
4. **Check distance constraints:** Accept a candidate x_{new} if it is at least $d_{\min}$ away from all existing points.
5. **Update the active list:** If a candidate is accepted, add it to the active list. If no valid candidates are found, remove x_{active} from the list.
6. **Repeat until the domain is filled:** Continue iterating until no active points remain.

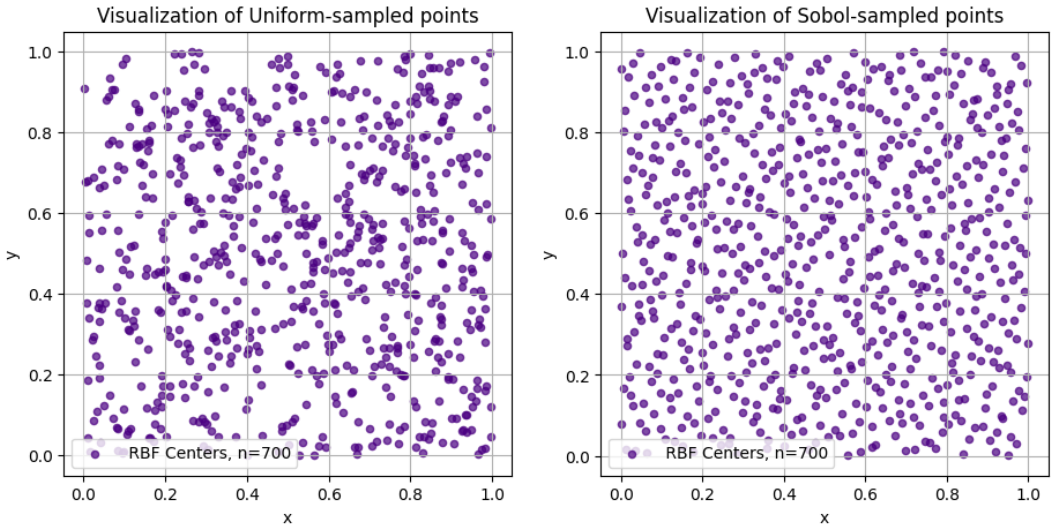

Figure 3.1: Sampled points with Uniform Sampling (left) and Sobol Sampling (right). The points in each plot can represent the RBF centers.

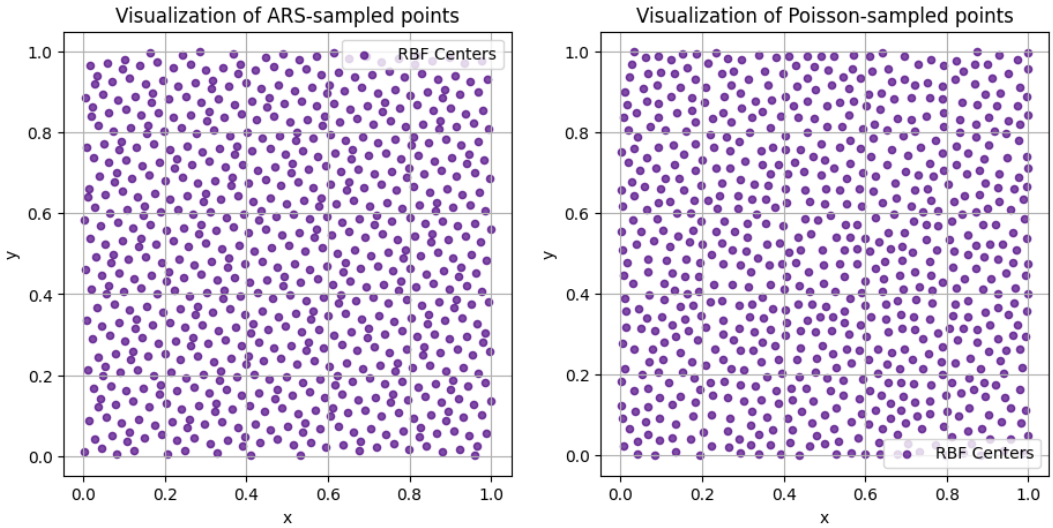

Figure 3.2: Sampled points with ARS (left) and Poisson Disk Sampling (right). The points in each plot can represent the RBF centers. We can observe that with these two methods the points are more evenly distributed than Sobol and Unform sampled points.

In Figure 3.2 we can observe an example of points sampled with Poisson Disk Sampling.

In the next chapter we analyze condition numbers and approximation accuracy of RBF and alternative basis functions, including B-Splines, Chebyshev polynomials, and Fourier Series to compare the performance of PIELMs architectures.

4 Analysis of Basis Functions to create efficient PIELM Poisson Solvers

This chapter focuses on the experimental study of different basis functions to construct PIELMs. The experiments aim to evaluate the condition number of different basis function methods and the performance of resulting PIELMs using each of those basis function methods. With investigating the impact of the amount of samples for a given domain and also the number of basis functions (or samples, in the RBF case) on the condition number and also the solution, we try to find the optimal combination to serve stability and accuracy with concrete type and amount of basis functions.

This study aims to provide groundwork for the extension of these methods to different problems, such as solving in higher dimensions or for time dependent PDEs, where the urge is to provide satisfactory models which are potentially able to scale to larger systems, crucial to solve real-world problems.

Observed is the impact of sampling methods introduced in Section 3.3, and the amount of basis functions per type on the condition number of the system matrix Q 4.9. Additionally, each basis function type, such as RBFs, Chebyshev polynomials, B-splines, Fourier Series, is examined to determine their impact on model accuracy, depending on the parameters, degree, amount of knot point and other features.

An analytical right-hand side of the Poisson equation is created using B-Splines, explained in Section 4.3.1. By systematically varying the setting, this chapter seeks to identify optimal configurations that improve the practical applicability of Physics-Informed ELMs. The results aim to contribute to the development of efficient and robust ELM-based solvers, advancing both theoretical understanding and practical implementation in computational physics.

Using robust numerical libraries like NumPy for matrix operations and JAX for differentiable programming, the implementation was carried out in Python. Data generation, model initialization, training, and evaluation were the main stages of the execution.

4.1 Motivation

Several studies proved that ELMs work well in combination with RBFs [23] [20] [37]. Nevertheless, there are still limitations, such as potentially high condition number of the RBF functions or the fact that uniformly sampled RBFs sometimes do not optimally fill the space, even if the solutions approximate very well and show good convergence.

For already providing high accuracy, an ELM RBF algorithm is already a good choice in terms of computational effort and error minimization, which gives groundwork to improve for even more efficient ways to deal with PDEs. The theory normally uses random sampling for the RBF network, hence the general architecture of an ELM also includes random choice of input weights. While this shows promising results, we want to try to investigate different possibilities of to use basis functions, which come with a higher value of stability for the condition number and also a faster error convergence. The aim of the further investigation

is to prove that other combinations of basis functions can improve stability and performance without fundamentally complicating the algorithm.

ELMs work well in combination with RBFs, but several limitations remain:

- **Condition Number Issues:** RBFs can lead to high condition numbers, which leads to the necessity of regularization, potentially degrading accuracy.
- **Sampling Challenges:** Uniformly sampled RBFs do not always optimally fill the domain.
- **Scalability Concerns:** As the number of RBF centers increases, computational efficiency decreases.

The goal is to find an optimal combination of basis function type and distribution that maintains accuracy while improving stability.

4.2 Evaluation Framework

The ongoing main objective is to describe, evaluate, and explain the numerical experiments that were carried out to evaluate how PIELMs performed when solving the Poisson equation. The purpose of the experiments described is to investigate how various setups and methods affect the accuracy, computational efficiency, and numerical stability of the solutions produced by the tested PIELMs.

The experiments to answer the main research topics were designed the following way:

1. **Impact of Sampling Strategies:** What effects do different domain sampling techniques have on the precision and stability of ELM solutions for the Poisson equation? This entails evaluating the impact of quasi-random sampling strategies, like Sobol sequences, on the model's performance by contrasting them with other strategies, such as Poisson Disk Sampling.
2. **Effect of Basis Functions:** What is the influence of different basis functions (e.g., Chebyshev polynomials, B-Splines) on the performance of ELM models? This helps evaluate the capacity of the ELM to capture and approximate complex function behaviors in the solution space.
3. **Evaluation of Boosting Techniques:** How do boosting strategies, which iteratively refine model predictions by learning residuals, compare to standard training approaches? The analysis seeks to determine whether iterative boosting can enhance solution accuracy, and whether this improvement justifies the additional computational effort.
4. **Parameter Sensitivity:** How do hyperparameters such as the number of hidden nodes or basis functions in the ELM network and (γ) for RBFs impact the model's overall performance? A sensitivity analysis is conducted to assess how these parameters influence the training process, model stability, and final solution accuracy.

To ensure a comprehensive evaluation of the Physics-Informed ELM's performance, we use several metrics:

- **Condition Numbers:** The condition number of the system matrix is analyzed to assess the numerical stability of the solution process. A high condition number indicates potential numerical instability, which could affect the model's ability to produce reliable results.
- **Error Metrics:** These include the L1 and L2-norm and the maximum error between the computed solutions and exact analytical solutions. The L2-norm provides an overall

measure of the model's accuracy by calculating the square root of the average squared deviations, while the maximum error highlights the greatest single-point deviation.

- **Computation Time:** Training and assessment procedures' time performance in various scenarios are investigated, with Just-In-Time (JIT) compilation.)

The outcomes will be covered in depth in the parts that follow, starting with the explanation of the analytical solution.

4.2.1 Implementation and Software

The implementation of the Physics-Informed Extreme Learning Machines (PIELMs) developed in this thesis was written in Python, utilizing the JAX library for automatic differentiation and optimized numerical computations. JAX was chosen due to its efficient handling of vectorized operations and its ability to leverage just-in-time (JIT) compilation, which significantly accelerates matrix operations and PDE solver performance. The framework allows seamless computation of derivatives, which is essential for applying the Laplace operator to the basis functions within the PIELM framework.

For the R-functions and implicit geometry representation, the MagPI library was employed. MagPI provides a robust implementation of Constructive Solid Geometry (CSG) through R-functions, allowing for precise encoding of complex domain geometries and boundary conditions directly within the function space. This integration ensures that boundary conditions are inherently satisfied within the solution representation, eliminating the need for explicit constraint enforcement in the numerical system [38].

The Extreme Learning Machine (ELM) architecture was also based on the MagPI framework, which provided the fundamental structure for randomized hidden layer weight initialization and fast analytical weight computation. This enabled the implementation of PIELMs without requiring iterative optimization methods, ensuring computational efficiency while maintaining numerical accuracy.

All experiments, including basis function evaluation, condition number analysis, and error computations, were performed using JAX-based implementations to maximize computational efficiency. The developed software allows for efficient numerical experimentation and provides a foundation for future research in machine learning-enhanced PDE solvers, particularly for micromagnetic applications and beyond.

4.3 Analytical Solution

To test our PIELM, we need to provide analytical solutions which will be approximated. An analytical solution requires an ansatz, i.e., a trial function that is assumed to have a specific structure but contains unknown parameters to be determined.

For Partial Differential Equations (PDEs), standard choices for an ansatz include:

- **Polynomial functions**: Useful for algebraic problems and simple differential equations.
- **Trigonometric functions (Fourier series)**: Commonly used in periodic problems and separation of variables.
- **B-Splines**: Used in computational analysis due to their smoothness properties and local support.

Once an ansatz $u(x)$ is chosen, it is substituted into the governing equation to check whether it satisfies the equation. For instance, the Poisson equation:

$$\Delta u(x) = f(x), \quad x \in \Omega, \tag{4.1}$$

now, with $\phi_i(x)$ being basis functions (e.g., B-Splines), then substituting into the equation gives:

$$\sum_i c_i \Delta \phi_i(x) = f(x). \tag{4.2}$$

In addition to satisfying the governing equation, the solution must also satisfy the boundary conditions, which depend on the physical or mathematical problem being solved. In the following experiments, boundary conditions are enforced using the hard constraint method explained in Section 3.2 or through a method to modify basis functions on a rectangular domain, which is shown in Section 4.3.1.

To ensure validity, we require that the basis functions $\phi_i(x)$ are differentiable up to the necessary order and that their linear combination spans a sufficiently large function space to represent $f(x)$.

We need to be careful that the selected ansatz does not automatically satisfy the equation, because in this case we cannot solve for any unknowns, making the problem trivial.

For example, if we would choose a linear ansatz for the homogeneous Poisson Equation:

$$u(x) = ax + b, \tag{4.3}$$

we compute the Laplacian:

$$\Delta u = \frac{d^2}{dx^2}(ax + b) = 0. \tag{4.4}$$

Since this ansatz automatically satisfies the equation, we cannot determine meaningful unknowns a and b, because they do not affect the validity of the equation.

4.3.1 B-Spline Solution

Our analytical solution for the test problems is constructed using B-Splines. It is created by a two-dimensional B-Spline representation on a chosen domain. The process begins by defining the *degree* of the B-Splines (in this case, cubic splines, $p = 3$) and a set of uniformly spaced knots within the interval $[-0.5, 0.5]$. To ensure smoothness at the boundaries, additional 'padding knots' are added at both ends, corresponding to the degree of the B-Spline. This ensures the continuity and differentiability of the spline up to the desired degree [7].

For a given input point $\mathbf{x} = (x_1, x_2)$, the B-Spline basis functions are evaluated independently for each dimension. These basis functions are constructed as:

$$u(\mathbf{x}) = \sum_{i,j} c_{ij} \cdot \phi_i(x_1) \cdot \phi_j(x_2), \tag{4.5}$$

where c_{ij} are the randomly sampled coefficients.[7]

The two-dimensional representation is obtained by computing the outer product of the basis functions in each dimension. A set of randomly generated coefficients c_{ij}, sampled uniformly, is associated with each basis function. These coefficients determine the 'weight' of each basis function in constructing the final solution. The analytical solution is then computed as the

weighted sum of all basis functions. Boundary conditions are satisfied either with the ADF 3.1.1 threw the hard constraint method or on the rectangle with changes applied to the B-Spline functions, which are shown in Figure 4.1.

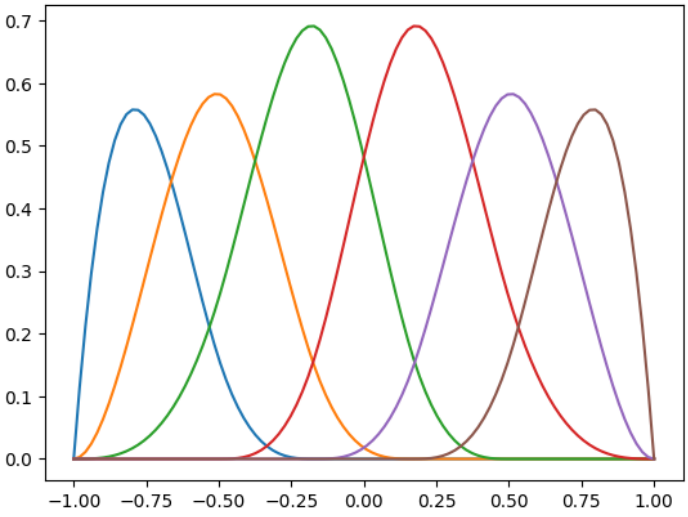

Figure 4.1: Representation of B-Splines in 1D with degree p=3, 4 knot points resulting in 6 B-Spline functions. To ensure that the boundary conditions are satisfied, "padding knots" are added at both ends and the interval for t is set to [-0.5, 0.5], which leads to the value 0 for all functions on the boundary.

4.4 Choice of ELM Basis Functions

In conventional ELM-based formulations, the system matrix Q 4.9 can become ill-conditioned when using randomly distributed RBF centers. This occurs because RBFs tend to overlap excessively in high-dimensional domains, leading to near-linear dependencies in the feature space and resulting in poor numerical conditioning when solving for output weights via the pseudoinverse.

Traditional RBF-based ELMs have been effective in function approximation due to their smooth nature. However, the ill-conditioning of the hidden layer matrix Q in large-scale PDE problems often results in numerical instability, making them less ideal for solving. This motivates the exploration of alternative basis functions, such as B-splines, which introduce compact local support, and Chebyshev polynomials, which enhance numerical stability via an orthogonal representation. By replacing RBFs with these alternative basis functions, we aim to improve the conditioning of the PIELM solver while maintaining model accuracy.

By converting the sampled input space into a functional representation, these basis functions act as the fundamental units of the solution. The model's correctness, stability, and computing efficiency are all greatly influenced by the basis function selection. Three types of basis functions are presented in detail in this section: *Radial Basis Functions (RBFs)*, *B-Splines*, and *Chebyshev Polynomials*.

One important indicator of numerical stability in Physics-Informed Extreme Learning Machines (PIELMs) is the condition number of the system matrix $\mathbf{Q}$. First, we are going to look at results from the evolution of the condition number of $\mathbf{Q}$ with different sampling methods, basis functions and increasing samples. Sobol, uniform, Poisson Disk and the Additive

Recurrence Sequence are among the sampling techniques taken into consideration.

To recap, The system matrix arises from applying the Laplace operator to the approximation in 2.23:

$$\Delta u_\beta(x) = \sum_{j=1}^{M} \beta_j \Delta \sigma_j(x). \tag{4.6}$$

From this, the system matrix Q is defined as:

$$Q_{ij} = \Delta \sigma_j(x_i), \tag{4.7}$$

Using the Singular Value Decomposition (SVD), we obtain Σ, which contains the singular values σ:

$$Q_{ij} = U \Sigma V^T \tag{4.8}$$

Finally, to obtain $\kappa(\mathbf{Q})$, we compute:

$$\kappa(Q) = \frac{\sigma_{\max}(Q)}{\sigma_{\min}(Q)} \tag{4.9}$$

For each basis function type, the condition number $\kappa(\mathbf{Q})$ was computed as the number of basis functions increased to 2^{10}. Now, that we introduced the method on how to obtain the condition number, the ongoing sections tackle the investigation of each basis function type for condition number and additionally error analysis.

4.5 Radial Basis Functions (RBFs)

As introduced in Section 3.3, Radial Basis Functions offer an efficient and accurate method to approximate functions in combination with ELMs. This section's aim is to discuss the conditioning of RBF functions, its applicability and potential improvements with better sampling techniques than Uniform Sampling.

4.5.1 Condition Number of RBFs with Uniform Sampling

In this section, we investigate the conditioning of and ELM that uses uniform sampled RBFs as basis functions, when we apply them to a specific problem. In this case, the Poisson Equation. If the results will show high condition numbers, we will try to use another mentioned sampling algorithm to improve the model. In depth results are shown in Chapter 4.

In Figure 4.3 and Figure 4.2 we can investigate that the condition number reaches very high values with the described setup which does not provide numerical stability. One assumption is that highly overlapping functions will result in poor conditioning and probably poorer results, and therefore a better method is attempted. One possibility is Poisson Disk sampling, which is described in 4.5. It allows a minimum distance to be introduced, which ensures a better distribution of the RBF centers.

4.5.2 Condition Number of RBFs with Poisson Disk Sampling

Figure 4.4 shows that the condition number stays constant over a higher number of samples, which is a good prerequisite for solving larger systems and also offers the possibility of sampling certain domains extremely precisely, if this is desired. Since the results for the

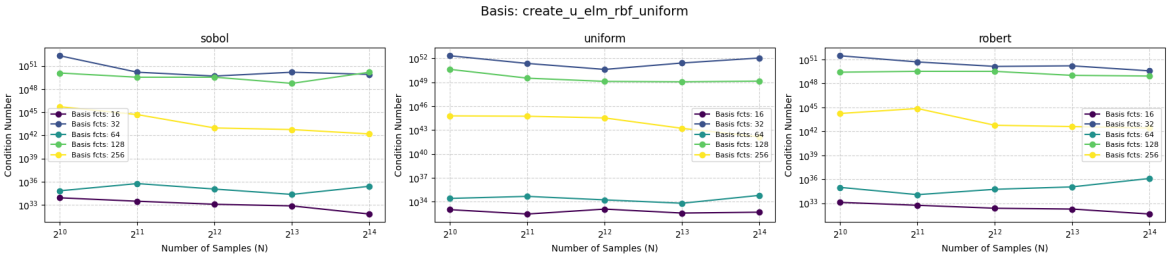

Figure 4.2: Plots of the Condition number of the system matrix Q 4.9 for the Poisson Equation for ELM with RBFs with increasing number of samples on the x-axis. The purple functions show results with 16 basis functions, green 128 and yellow 256. With domain sampling methods (left to right) Sobol, Uniform and Robert (Additive Reccurrence) sampling. Results show that the condition number stays constant with higher number of samples.

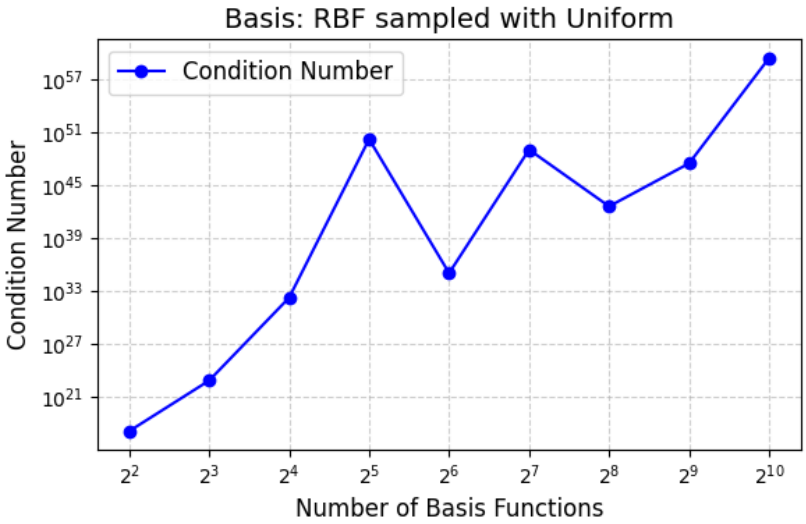

Figure 4.3: Condition number of the solution operator matrix (Q) for ELM with RBFs with increasing number of basis functions. RBFs are sampled with uniform sampling, gamma = 20 and the test domain used was Sobol with 10^{13} samples.

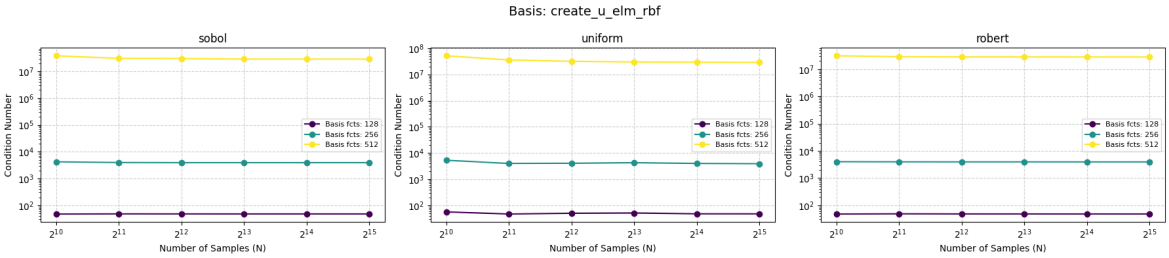

Figure 4.4: Plots of the Condition number of the system matrix Q 4.9 for ELM with Poisson Disk samples RBFs with increasing number of samples on the x-axis. The purple functions show results with 128 basis functions, cyan 256 and yellow 512. With domain sampling methods (left to right) Sobol, Uniform and Robert (Additive Reccurrence) sampling. Results show that the condition number stays constant with increased samples.

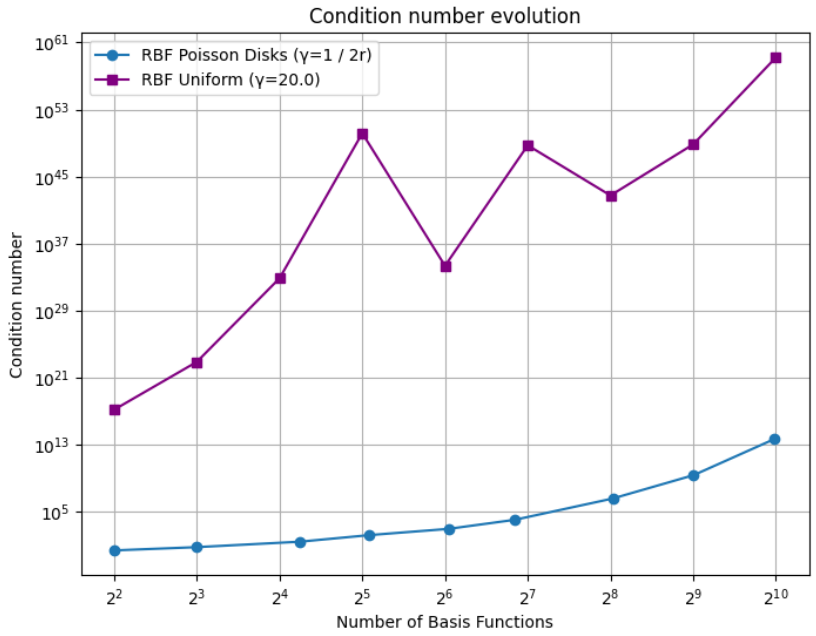

Figure 4.5: Condition number of Q 4.9 for both ELM RBF methods compared with each other, with Poisson Disk gamma = 1 / 2r being adaptive and fixed for the uniform RBF. Note that a regularization term of $10^{-5}$was introduced to balance the method.

amount of basis functions are not completely clear yet, we want to further investigate it with a concrete condition number plot for an increasing number of basic functions.

We can observe in Figure 4.5 a significant decrease in the growth of the conditional number when using the new Poisson Disk RBF model with a specific *gamma* parameter, that could be a very efficient combination to not only be better-conditioned but also providing satisfactory approximations for our problem, since the sampling method is incremental and also the RBF combination is already proven to be a good universal approximator [32].

4.5.3 Error Analysis

Furthermore, the different RBF setups are examined to investigate improvements from the Poisson Disk sampling method. Results show that the new method seems to minimize the error even better than our initial configuration. Both L1 and L2-Errors are investigated.

With analyzing the results in Figure 4.6, we observe that the model performs well, demonstrating satisfactory accuracy and faster error convergence. Furthermore, the use of Poisson Disk Sampling for placing RBF centers significantly improves accuracy compared to uniform sampling. This is evident from the error convergence plots in Figure 4.6, where the Poisson Disk approach consistently yields lower errors across rise of basis functions.

The improvement can be attributed to the structured placement of RBF centers in Poisson Disk Sampling, which ensures a more uniform coverage of the domain while preventing excessive clustering. This leads to better function representation and a better conditioned system matrix, reducing numerical instability.

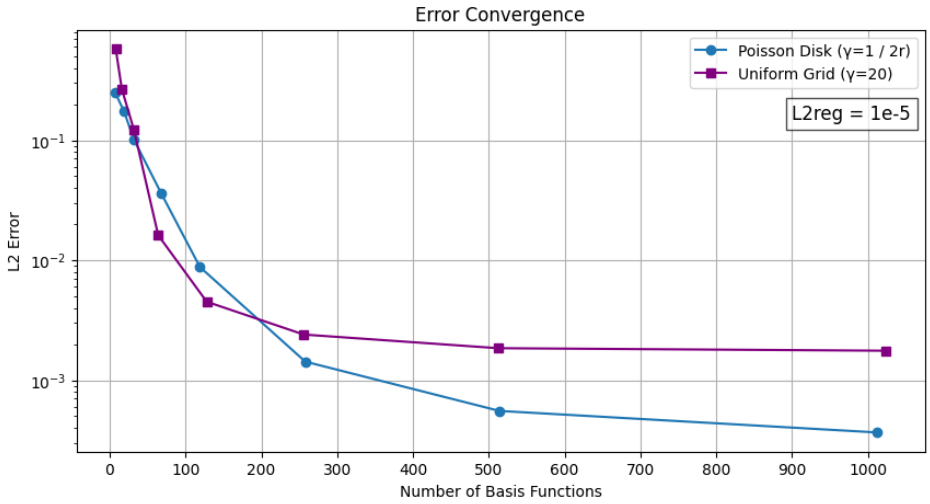

Figure 4.6: ELM-RBF L2-error convergence for both RBF methods with increasing number of basis functions. Error convergence for both RBF methods with increasing amount of basis functions from 2^5 to 2^{11}. Approximated solution is the solution from 4.3.1. Observable is that the error converges similarly for both configurations, with advantage to the Poisson Disk method.

4.6 B-Splines

B-Splines (Basis Splines) are piecewise polynomial functions defined over a set of knots. They are widely used in numerical analysis due to their local support, flexibility.

A B-Spline of degree p is defined recursively:

Base Case (Degree $p = 0$):

$$B_{i,0}(x) = \begin{cases} 1, & t_i \leq x < t_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

Each function is nonzero only within a single interval, making it discontinuous at knots.

Recursive Formula for Higher Degrees ($p \geq 1$)

$$B_{i,p}(x) = \frac{x - t_i}{t_{i+p} - t_i} B_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(x), \tag{4.11}$$

where t_i are the knot positions, forming a non-decreasing sequence:

$$t_0 \leq t_1 \leq \cdots \leq t_m. \tag{4.12}$$

4.6.1 Degree and Smoothness of B-Splines

The degree p of the B-Spline controls its smoothness at the knot points:

Degree p	**Function Type**	**Continuity at Knots**	**Differentiability**
$p = 0$	Piecewise constant	Discontinuous	Not differentiable
$p = 1$	Piecewise linear	Function continuous	1st derivative discontinuous
$p = 2$	Piecewise quadratic	1st derivative continuous	2nd derivative discontinuous
$p = 3$	Piecewise cubic	2nd derivative continuous	3rd derivative discontinuous

Cubic B-Splines ($p = 3$) are prominent in numerical applications as they provide a smooth, twice continuously differentiable representation while maintaining local support.

For 2D domains, the B-spline basis functions are constructed as:

$$\phi_{ij}(\mathbf{x}) = B_i(x_1)B_j(x_2). \tag{4.13}$$

Key Properties:

- *Local Support:* Each B-spline is nonzero only within a limited range of the knots, making them computationally efficient.
- *Smoothness:* The smoothness of the spline depends on its degree p. For example, cubic splines ($p = 3$) are twice continuously differentiable.
- *Adaptability:* The placement of knots allows control over the resolution and density of the basis functions.

In Figure 4.1 we can observe that the starting and end points of the B-Splines were modified in the underlying splines in order to satisfy the boundary conditions.

As these linear splines expect to have a better condition number, we do not exactly know if they will also be able to approximate the solution in a great sense. To ensure smoothness at the boundaries, additional "padding knots" are added at both ends, this ensures that the boundary conditions are satisfied.

By choosing cubic B-Splines we want to achieve a well-conditioned numerically stable basis suitable for solving the Poisson equation efficiently. First, we will investigate again the condition number of the solution operator with B-Splines.

4.6.2 Condition Number

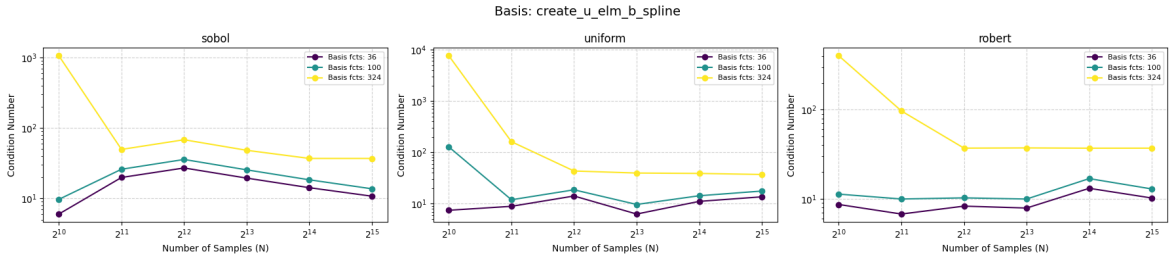

Figure 4.7: Plots of the Condition number of Q 4.9 for ELM B-Splines degree=3 with increasing number of samples on the x-axis. The purple functions show results with 36 basis functions, cyan 100 and yellow 324. With domain sampling methods (left to right) Sobol, Uniform and Robert (Additive Reccurrence) sampling. Results show that the condition number stays constant with higher number of samples.

Figure 4.7 shows that also here the conditioning number remains constant with a high number of samples. Observable is that the condition number remains small, also with higher amount of basis functions. In a next test, we investigate the characteristic of the condition number with increased basis functions, which could maintain stable and thus provide a potentially very useful method to solve PDEs.

The plot in Figure 4.8, which illustrates the condition number as a function of the number of basis functions, reveals an important trend: The condition number stays very constant, even with strong increase of basis functions. This suggests that while increasing the number of basis functions enhances the accuracy of the approximation, it simultaneously does not

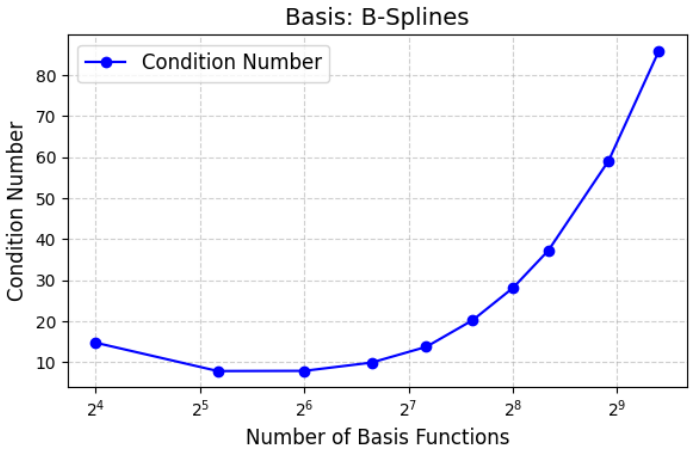

Figure 4.8: Condition number of the system matrix Q 4.9 for ELM with B-Splines with increasing number of basis functions. B-Splines with p = 3 and are constructed with increasing knot points, with t element of [-1, 1].

degrade the numerical stability of the system matrix Q 4.9, which is a positive feature. In practical applications, this means that the amount of basis functions with B-Splines can be chosen in much higher amounts than for example RBFs, with still maintaining a very low condition number.

4.6.3 Error Analysis

Since we can observe in Figure 4.8 a constant behavior of the condition number, we want to analyze the absolute error of the approximated solution using B-Splines in the network.

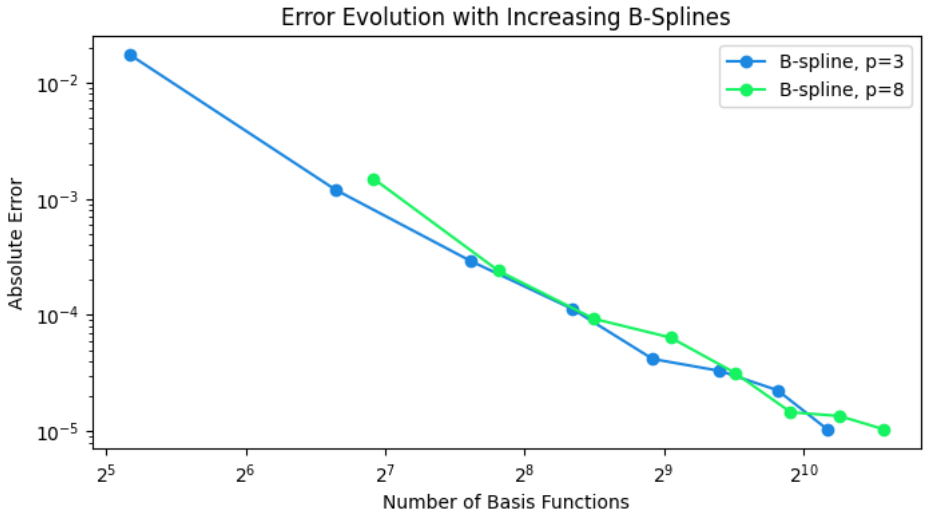

Figure 4.9: Error convergence of B-Spline ELM with degree p=3 and p=8 with increasing amount of knot points (= basis functions) from 2^5 to 2^{11}. Approximated solution is the solution from 4.3.1. Observable is that the error converges similarly for both configurations, showing that p=3 is also satisfactory.

Analyzing the results of Figure 4.9 with keeping in mind that the condition number is small,

these results are promising. First, we can observe that the actual degree of the B-Splines shows very similar results with also lower degrees, which enables us to use the smallest degree necessary to save computational effort. The comparison of the errors in the Table 4.3 shows the similar performance of B-Splines and other PIELM methods, whereby we point out that the B-Spline method provides satisfactory results, taking also the condition number into account. Moving on to the next method, we keep in mind that we already achieved good results with B-Splines. B-Splines mitigate this issue by ensuring that each basis function is compactly supported, meaning that it contributes to function approximation only within a limited region around its assigned knots. This localized influence prevents basis function redundancy and reduces numerical instability when solving structured PDEs.

For solving the Poisson equation in micromagnetics, these findings are particularly relevant. The Poisson equation governs the magnetostatic potential in micromagnetic models, where smooth and accurate field approximations are essential. The use of B-Splines is advantageous because they provide smooth and stable approximations.

4.7 Chebyshev Polynomials

Chebyshev polynomials are a family of orthogonal polynomials defined on the interval $[-1, 1]$. They are widely used in numerical methods due to their efficient approximation properties. The n-th Chebyshev polynomial of the first kind is given by:

$$T_n(x) = \cos(n \arccos(x)), \quad x \in [-1, 1]. \tag{4.14}$$

Key Properties:

- *Orthogonality:* Chebyshev polynomials are orthogonal with respect to a specific weight function, ensuring minimal redundancy in the basis representation. Note that with multiplying with ADFs - shown in Section 3.1.1 - we lose the orthogonality property.
- *Efficient Approximation:* Their smoothness and global nature make them well-suited for approximating solutions with global properties.
- *Numerical Stability:* The use of trigonometric definitions provides good numerical stability when evaluating higher-order polynomials.

Chebyshev polynomials provide a spectral approximation of the solution, leveraging orthogonality properties to minimize error accumulation while hopefully maintaining a well-conditioned system matrix. This structured basis potentially reduces the number of required neurons while likely ensuring high accuracy.

4.7.1 Condition Number

Figure 4.10 shows similar behavior than with other basis functions, that the amount of samples is not decisive. With further investigation, the plot in Figure 4.11 shows, that the condition number operates in a range that is still considerably smaller than that of the RBFs. Nevertheless, the increase is not as good as with the B-splines.

Figure 4.11 illustrates how the condition number evolves as the number of Chebyshev basis functions increases. Initially, the condition number remains relatively low and constant, suggesting that for a moderate number of basis functions, the numerical system remains well-conditioned. However, as the number of basis functions increases beyond 2^9, the condition number exhibits an abrupt and strong growth, indicating a slight decrease in numerical stability.

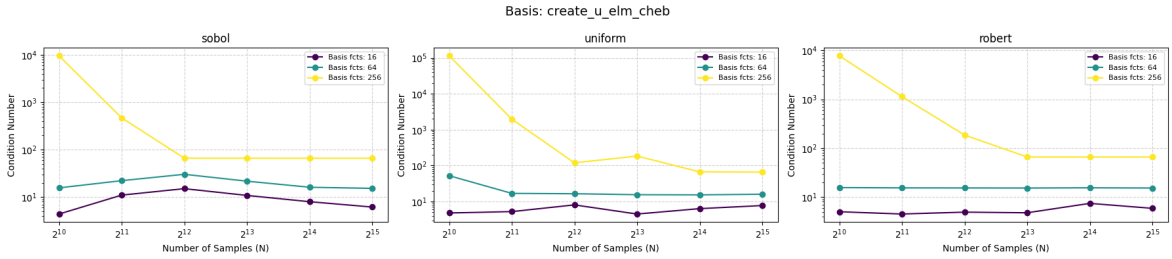

Figure 4.10: Plots of the Condition number of Q 4.9 Matrix for ELM with Chebyshev polynomials with increasing number of samples on the x-axis. The purple functions show results with 16 basis functions, cyan 64 and yellow 256. With domain sampling methods (left to right) Sobol, Uniform and Robert (Additive Reccurrence) sampling. Results show that the condition number stays constant with higher number of samples and remains small also for 256 basis functions.

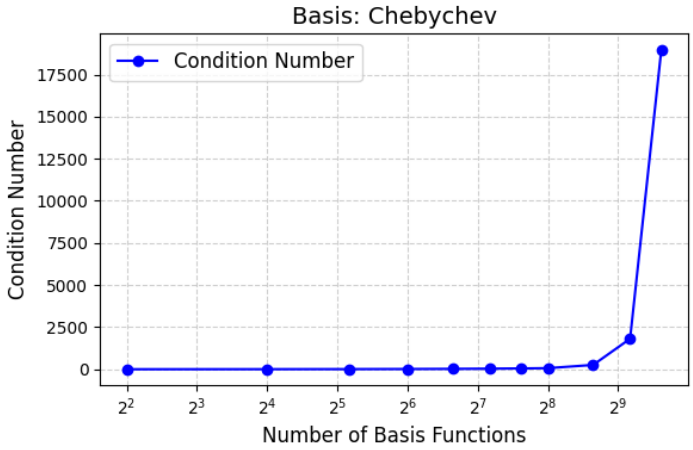

Figure 4.11: Condition number of the matrix Q 4.9 for ELM with Chebyshev polynomials with increasing number of basis functions. Chebyshev polynomials operate between [-1, 1]. At an amount of 2^9 basis functions, we can observe a significant increase of the condition number.

This behavior is expected due to the nature of Chebyshev polynomials. While they possess very good approximation properties due to their orthogonality and optimal interpolation characteristics, their numerical stability degrades at higher polynomial degrees. This effect is linked to the Runge phenomenon, where high-degree polynomials introduce oscillations that amplify numerical errors, leading to poor conditioning of the system matrix Q 4.9 [1].

From an application perspective, this implies that while Chebyshev polynomials are highly effective in capturing smooth variations in the magnetostatic potential, their use must be carefully managed when dealing with large basis function counts. To mitigate this, preconditioning techniques, regularization, or limiting the number of Chebyshev polynomials to a stable range could help maintain numerical robustness. For highly localized magnetic structures, alternative basis functions like B-Splines or RBFs with localized sampling may be more stable and reliable.

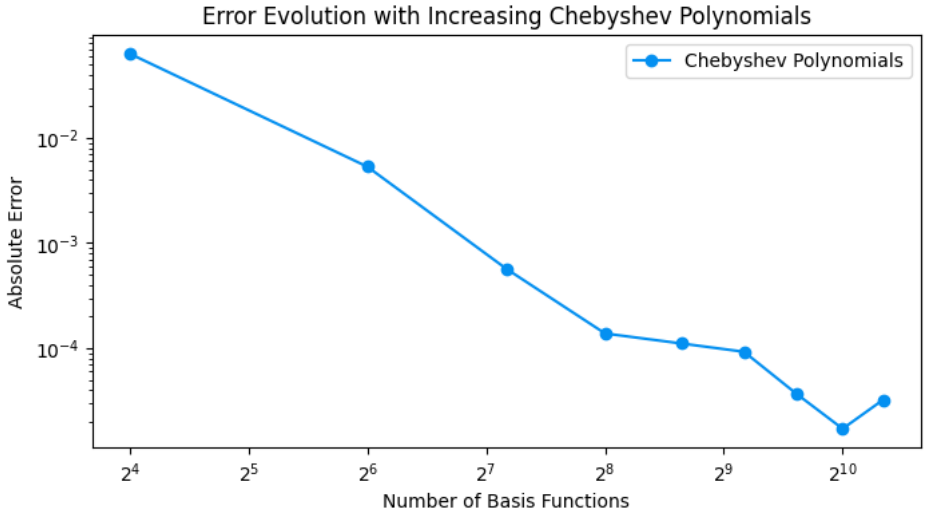

Figure 4.12: Error convergence of Chebyshev ELM with degree with increasing amount of basis functions from 2^5 to 2^{11}. Approximated solution is the solution from 4.3.1, showing that the error decreases continuously until a slight change of the trend when the number of basis functions exeeds 2^{10}.

4.7.2 Error Analysis

Figure 4.12 shows that the absolute error converges around 10^{-4} with approximately 2^8 basis functions. A clear downward trend in error is observed as the number of basis functions increases, demonstrating a good accuracy potential of Chebyshev polynomials in approximating the solution of the Poisson equation.

However, at the largest basis function counts, a slight increase in error appears after reaching a minimum. This confirms the previously observed conditioning issues: As the basis functions reach a peak, numerical errors start to propagate, slightly reducing the accuracy. This highlights a practical consideration: in this setup there probably exists a peak for the number of Chebyshev polynomials where accuracy is maximized until the error slightly increases again.

4.8 Fourier Series

Fourier series provide a spectral approach to function approximation by decomposing a function into a sum of sine and cosine terms. They are particularly well-suited for representing periodic functions but can also be adapted for general function approximation with appropriate modifications.

The general form of a Fourier series expansion of a function $f(x)$ is:

$$f(x) = a_0 + \sum_{n=1}^{M} a_n \cos(n\pi x) + b_n \sin(n\pi x), \tag{4.15}$$

where:

- a_n and b_n are the Fourier coefficients that determine the contribution of each basis function,
- n represents the frequency component,

- πx ensures periodicity in the interval of definition.

Key Properties:

- **Global Representation**: Unlike localized basis functions such as B-splines or RBFs, Fourier basis functions span the entire domain.
- **Orthogonality**: The sine and cosine terms form an orthogonal basis under the standard inner product, ensuring stable numerical computations. Note that with multiplying with ADFs - shown in Section 3.1.1 - we lose the orthogonality property.
- **Spectral Convergence**: Fourier series exhibit rapid convergence for smooth functions, making them efficient for high-accuracy approximations.
- **Boundary Condition Handling**: Fourier basis functions naturally satisfy periodic boundary conditions, but modifications (such as multiplying by weighting functions) can enforce Dirichlet or Neumann boundary conditions.

For solving PDEs such as the Poisson equation, Fourier basis functions can also be combined with appropriate constraints to ensure the solution adheres to required boundary conditions. This makes them a powerful tool for problems with structured or symmetric domains.

4.8.1 Condition Number

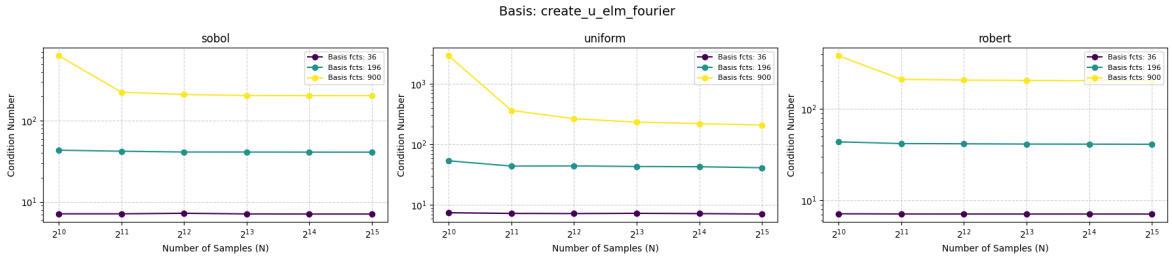

Figure 4.13: Plots of the Condition number of Q 4.9 for ELM Fourier Series with increasing number of samples on the x-axis. The purple functions show results with 36 basis functions, cyan 196 and yellow 900. With domain sampling methods (left to right) Sobol, Uniform and Robert (Additive Reccurrence) sampling. Results show that the condition number stays constant with higher number of samples and remains small for even 900 Fourier Series functions.

Similar to B-Splines, we observe in Figure 4.13 a constant function with increase of samples. The evaluation of the basis function amount will show further insight about the conditioning of this method.

Figure 4.14 shows how the condition number evolves as the number of Fourier basis functions increases. Initially, the condition number remains very low, suggesting that Q 4.9 is well-conditioned when using a smaller number of basis functions. As the number of basis functions increases, the condition number stays quite constant, similar to B-Splines. Now we want to observe the error evolution of this method, hoping that it will also show a continuous decrease.

Note the fact that the ADF, explained in R-Functions Section 3.1.1, is necessary to handle the shape of the domain well and with it we loose orthogonal properties. Unlike polynomials, which tend to introduce instability at high orders, Fourier basis functions maintain relatively stable numerical properties as long as the function being approximated is smooth and periodic.

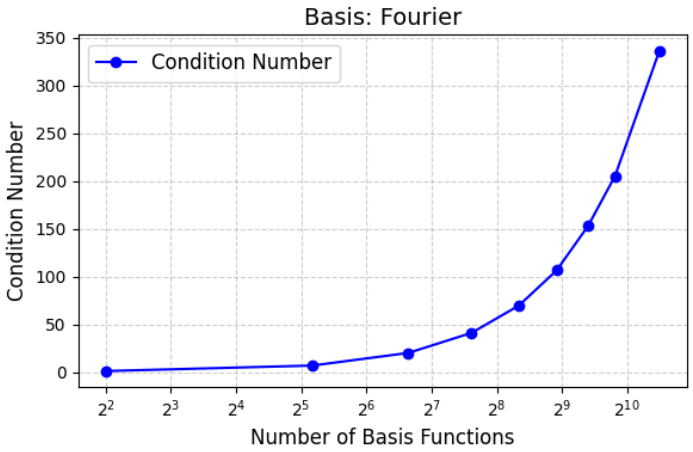

Figure 4.14: Condition number of the system matrix Q 4.9 for ELM with Fourier Series with increasing number of basis functions. We can observe a similar behavior as with B-Splines, that the condition number remains quite constant, which is advantageous for example if we want to apply iterative solvers.

4.8.2 Error Analysis

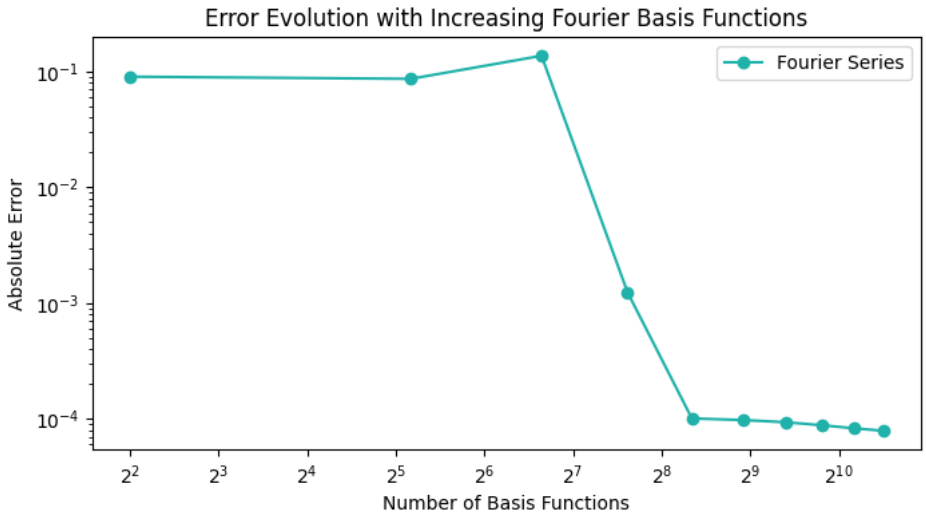

Figure 4.15: Error convergence of Fourier Series ELM with increasing amount of basis functions from 2^5 to 2^{10}. A ridge regularization term of 1e-4 was implemented to stabilize the approximation. Approximated solution is the solution from 4.3.1 showing that the error decreases after reaching a certain threshold.

Interesting to observe in Figure 4.15 is the sudden decrease of the error, with the approximation to reach stableness after a certain amount of basis functions used. The regularization term avoids fluctuations and inaccuracies with higher amounts of basis functions. The trend reveals a sudden drop in error after an initial plateau, indicating that Fourier series require a minimum number of basis functions before they can effectively approximate the solution. After reaching this threshold, the error decreases rapidly and stabilizes at a low level, demonstrating that Fourier Series do not perform extremely well with this problem. The issue that a certain threshold is needed to approximate the solution well makes this approach less

practicable, if considered that a threshold needs to be examined, which will likely vary in different problem settings.

4.9 Comparison

4.9.1 Condition Number

Clearly observable is the fact that the condition number stays constant at a certain amount of samples and further increasing the domain samples does not have any other noticeable impact on the evolution of the condition number. Therefore, we do not have to worry about 'oversampling' the domain in terms of stability. Also, the condition number stays constant as the number of samples rises, regardless of the sampling technique employed, which indicates that the numerical stability of the basis functions does not strongly depend on the domain sampling method. Even uniform domain sampling, which is subject to artifacts in smaller data sets, maintains a stable condition number as the sample size increases.

B-Splines and Fourier Series exhibit a desirable property where the condition number stays also constant with a higher number of basis functions. This relationship indicates a well-conditioned system as the complexity increases. In contrast, Chebyshev polynomials and RBFs demonstrate a stronger growth in the condition number as the number of basis functions increases.

Basis Function	**CN (2^8 BFs)**	**CN (2^{10} BFs)**	**Growth Behavior**
RBF (Poisson Disk)	10^5	10^{13}	Very Strong growth
B-Splines	10^1	10^2	Constant (or minimal growth)
Chebyshev Polynomials	10^2	10^5	Moderate growth
Fourier Series	10^1	10^2	Constant (or minimal growth)

Table 4.1: Condition number comparison for Q 4.9 of different basis functions. CN and BF denoted here as 'Condition number' and 'Basis functions'.

In Figure 4.1, a concise comparison shows the different characteristics of every investigated basis function type for the condition number of system matrix, Q 4.9. With B-Splines and Fourier Series showing very good results and Chebyshev satisfactory results. Since Table 4.1 shows that for our problem, the number 2^8 for the basis functions seem to be a good trade-off for condition number and quality of approximation, all methods are applicable. A further comparison will observe how this methods perform with many different analytical solutions. The ultimate goal is to design PIELMs that achieve the best trade-off between numerical stability, computational time, and solution accuracy.

4.9.2 Error Analysis

The results in the Tables 4.3, 4.4 and 4.5 are computed by a method that evaluates the accuracy of different basis functions by repeatedly training a model and comparing its results to 1000 different analytical solutions on different shapes. It starts by initializing test samples and a global random key to ensure reproducibility. In each iteration, the exact solution is computed for a test domain, and a model is trained to approximate it. The error between the model's output and the analytical solution is then calculated and stored. All errors are saved and only averaged once at the end. To introduce controlled randomness, the global key is updated at each step. Finally, the total mean error is computed by averaging all stored errors, providing a clear measure of how well the basis functions approximate the solution.

As the number of basis functions rises for the B-Spline, Chebyshev and RBF methods the absolute error rapidly decreases. Except for the Fourier Series, which need to arrive at a certain threshold, the error with the other methods decreases sharply, suggesting that the quality of the approximation is greatly enhanced by the addition of more basis functions. Given that a larger number of basis functions enables the model to capture more intricate aspects of the solution, this tendency is to be expected. With a lower number of basis functions, B-Splines have a slight advantage over Chebyshev polynomials, they achieve a lower absolute error for the number of basis functions.

Although increasing the number of basis functions increases accuracy, the plateau in error reduction suggests importance of identifying the 'optimal' number of basis functions for a given problem in order to avoid unnecessary computational overhead, which is visible at about 256 basis functions for the tested methods.

Basis Function	**Absolute Error (2^8 BFs)**	**Absolute Error (2^{10} BFs)**
RBF (Poisson Disk)	2×10^{-3}	10^{-3}
Chebyshev Polynomials	10^{-3}	10^{-4}
B-Splines	10^{-4}	10^{-5}
Fourier Series	10^{-3}	10^{-4}

Table 4.2: Absolute error comparison for a single random analytical solution 4.3.1 with different basis functions for 2^8 and 2^{10} samples. BFs denoted as 'basis functions'

In Table 4.2 the results and errors for a single random analytical solution with different basis functions are presented.

To further analyze the accuracy and performance of these functions, the following tasks are executed to obtain results of iteratively approximated solution to investigate mean time and error with 1000 different analytical solutions on different domain shapes. Time is measured before and after the full task for each method independently to create comparable results, which includes the calculation of the Singular Value Decomposition (SVD). Therefore, the computation time is more an estimate of the preparation of the actual computation, because the computation itself, apart from the SVD appears to be much smaller.

Figures 4.16, 4.17, 4.18 show examples of approximations of randomly generated analytical solutions on different geometries. Domain shapes of rectangle, cube and cutted domain are ensured via R-functions. The error estimation was made on these three domains, each with 1000 different analytical solutions, to obtain a mean error and standard deviation for each PIELM solver.

4.9.2.1 Rectangle Domain

Results in Table 4.3 for the rectangle domain present that the overall best performance with 2^8 basis functions is done with Chebyshev Polynomials, with having a slightly smaller error than even B-Splines. The results presented in the table provide a comparison of different basis functions regarding both accuracy and computational time (including the SVD), in solving the Poisson equation on a rectangle using 2^8 basis functions. The mean absolute error (MAE) values indicate that Chebyshev polynomials achieve the highest accuracy, with an error of 0.000123, followed closely by B-Splines (0.000158). Radial Basis Functions (RBFs) with Poisson Disk Sampling exhibit the highest error, reaching 0.000575.

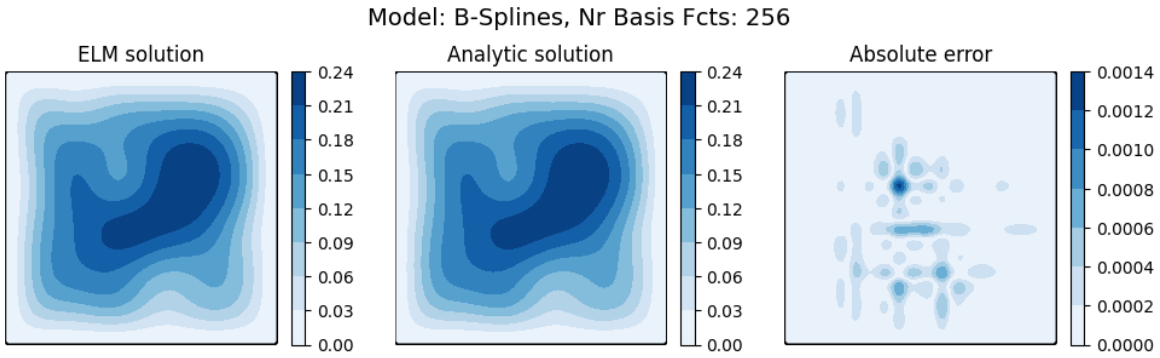

Figure 4.16: An example comparison of ELM solution, analytical solution, and absolute error on rectangle. Approximated with B-Splines. The black lines represents the boundary.

Basis Function	**MAE on Rectangle - (2^8 BFs)**	**Mean Time (s)/Computation**
RBF (Poisson Disk)	0.000575 ± 0.000123	0.003935
Chebyshev Polynomials	0.000123 ± 0.000049	0.003544
B-Splines	0.000158 ± 0.000044	0.003191
Fourier Series	0.000272 ± 0.000922	0.004095

Table 4.3: Comparison of mean absolute error (MAE) and computation time for different basis functions (BFs) of 1000 analytical solutions 4.3.1 on a rectangle.

4.9.2.2 Circle Domain

Results shown in Table 4.4 that again show high accuracy of Chebyshev Polynomials. Also B-Splines show very good results. The MAE for Chebyshev polynomials shows the highest accuracy, with an error of 0.000240, followed closely by B-Splines (0.000315).

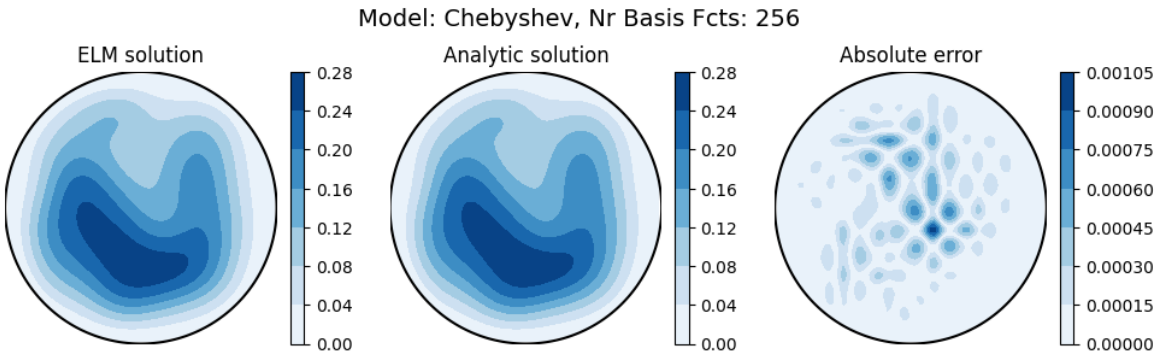

Figure 4.17: Example comparison of ELM solution, analytical solution, and absolute error on a circle. Approximated with Chebyshev polynomials. The black line represents the boundary.

Basis Function	MAE on Circle - (2^8 BFs)	Mean Time (s)/Computation
RBF (Poisson Disk)	0.001045 ± 0.000236	0.002831
Chebyshev Polynomials	0.000240 ± 0.000047	0.002264
B-Splines	0.000315 ± 0.000083	0.002169
Fourier Series	0.000307 ± 0.001024	0.003466

Table 4.4: Comparison of mean absolute error (MAE) and computation time for different basis functions (BFs) of 1000 analytical solutions 4.3.1 on a circle.

4.9.2.3 Cutted Domain

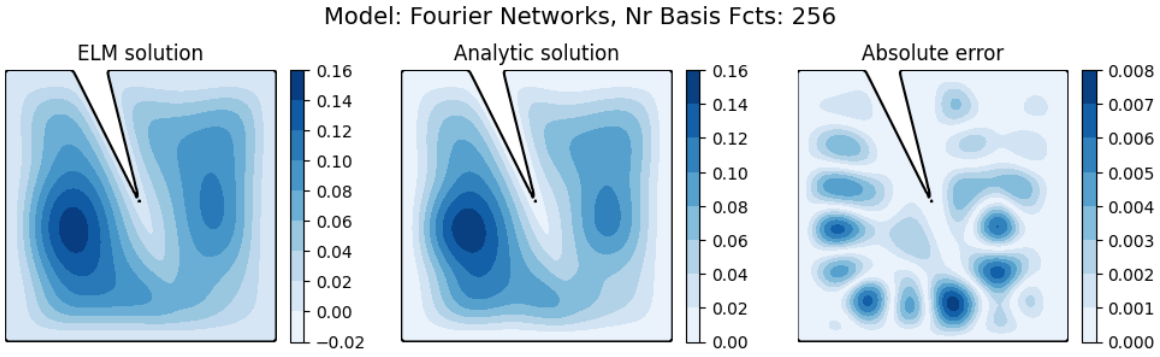

Figure 4.18: Example comparison of ELM solution, analytical solution, and absolute error on a cutted domain. Approximated with Fourier basis functions. The black line represents the boundary.

Results shown in Table 4.5 for the cutted domain reveal B-Splines with the smallest error of 0.000185, followed by Chebyshev Polynomials. Overall, all tested methods show good results for this domain.

Basis Function	MAE on Domain X - (2^8 BFs)	Mean Time (s)/Computation
RBF (Poisson Disk)	0.000174 ± 0.000031	0.004666
Chebyshev Polynomials	0.000216 ± 0.000067	0.004029
B-Splines	0.000185 ± 0.000048	0.003549
Fourier Series	0.000240 ± 0.000720	0.004183

Table 4.5: Comparison of mean absolute error (MAE) and computation time for different basis functions (BFs) of 1000 analytical solutions 4.3.1 on a cutted domain.

Chebyshev polynomials appear to be the best choice for high accuracy. However, their condition number growth at high function counts suggests that their usage should be carefully managed to prevent numerical instability. For large-scale simulations, B-Splines considerably offer a good balance between number of basis functions and accuracy. Their well-conditioned system matrices make them an attractive choice for future applications. Although RBFs provide flexibility in handling complex geometries, in this direct comparison they perform slightly less accurate than B-Splines and Chebyshev polynomials. Fourier Series overall performed with the lowest accuracy. Extending this analysis to three-dimensional simulations would provide further insight into the scalability of these basis functions. Furthermore, integrating boosting techniques into PIELMs, such as adaptive boosting or gradient boosting, could improve convergence and numerical robustness.

5 Boosting

To improve the performance of the initial model, a gradient boosting strategy was implemented. This approach involved training a second ELM model on the residuals of the initial model. The residuals were calculated as the difference between the Laplacian of the model predictions and the Laplacian of the true solution. This second model aimed to correct the errors of the initial one by learning from its shortcomings.

At each iteration, the output of the second model is added to the predictions of the first model, scaled by a learning rate parameter. This boosting process continued iteratively, refining the model with each step and improving its overall performance. The residuals were recalculated at each iteration, and the process was repeated until convergence was achieved or a predetermined number of iterations was reached [11, 19].

The iterative boosting strategy was essential to capture higher-order features and improve the accuracy of the in regions where the initial model performed worse. The effectiveness of this approach was monitored by observing the L2 error at each boosting step and tracking how the error converged over the iterations [16].

This section deals with the application of Gradient Boosting for the accuracy of an Extreme Learning Machine based on the solving of the Poisson equation. The basic ELM was designed to approximate the solution of the equation with high efficiency, and the continued work has focused on iteratively reducing errors to refine its predictions. This section describes the experiments conducted, the methodology implemented, and the results obtained, providing detailed information on the process and results.

5.1 Gradient-Boosted PIELM Model

By training new models on the residual errors of the prior model, Gradient Boosting was used to improve the model predictions. The following is a description of the method.

At each boosting iteration, the residual $r(x)$ was computed as the difference between the Laplacian of the true solution and the Laplacian of the model's predictions:

$$r(x_i) = \Delta u(x_i) - \Delta u_{\text{model}}(x_i). \tag{5.1}$$

A new ELM was trained to minimize the residuals using the same architecture and methodology as the baseline model. Specifically, the hidden layer output $u_{\text{elm, res}}(x)$ was computed, and the corresponding weights β_{res} were determined by solving:

$$\beta_{\text{res}} = Q_{\text{res}}^{+} r, \tag{5.2}$$

where Q_{res} is the Laplacian of the hidden layer outputs for the new model.

The updated model was expressed as a sum of the previous model and the new model weighted by a learning rate α:

$$u_{\text{model, updated}}(x) = u_{\text{model}}(x) + \alpha u_{\text{elm, res}}(x). \tag{5.3}$$

[11]
The L_2-error of the updated model was computed after each iteration:

$$L_2 = \sqrt{\int_\Omega \left(u_{\text{model}}(x) - u(x)\right)^2 \, dx}. \tag{5.4}$$

5.2 Results

The application of gradient boosting for PIELMs brought to light a number of issues as well as some interesting findings. The model's complexity comes with great calculation needs. Optimizing a boosting procedure could greatly lower computational overhead and increase scalability. Another difficulty was to achieve fast convergence, which was obtained after bunch of tests and tuning the Poisson Disk radius and gamma parameters. Although the error reduction was significant in the early stages, the improvements tapered off as the error approached a lower bound. This behavior is also attributed to the fact that the baseline ELM already achieved a relatively low error.

An important observation was the impact of the learning rate α on the acceleration process. While a smaller α slowed convergence, it provided greater stability and smoother error reduction, whereas larger values occasionally introduced oscillations in the error evolution. For this experiment, a value of $\alpha = 1$ was chosen, which balanced convergence speed and stability effectively. Even though the boosted model performed much better than the baseline ELM, the additional ELMs trained on residuals and the iterative nature of boosting increased its computing cost. The method is computationally more costly than the baseline model since each iteration required the development of a new model and the recalculation of the Laplacian operator. These difficulties show that accuracy and processing efficiency must be carefully balanced, especially when applying the method to higher-dimensional or more complicated issues.

The RBF activations' spread was also significantly impacted by the scaling factor γ. The model demonstrated a superior fit to the training data, resulting in reduced L2 errors, when γ was appropriately selected. Conversely, excessive values of γ either led to improper convergence or overfitting of the model. The significance of choosing the right γ value to balance model performance and training stability was highlighted by this tests.

Tests were made with the RBF Poisson Disk model, since the sampling also provides an iterative scheme that matches perfectly with the idea of continuously creating new models with small amount of basis functions. The accuracy of the model was significantly improved by the Gradient Boosting procedure, especially in the early iterations. Figure 5.1 shows how the L_2-error changed over 50 iterations. Over the course of boosting, the baseline ELM's initial error, which was roughly 0.02, decreased. The error had already decreased to 0.005 by iteration 10, and stayed almost stable until iteration 50. Also interesting was the fact that with the small amount of basis functions also the condition number was very small and stable between $300 - 1000$.

The results for the boosted model are shown in Figure 5.2. By effectively reducing the high-error areas found in the baseline model, the Gradient Boosting procedure produced a more consistent error distribution throughout the domain. The boosted model captured finer details of the solution, with the majority of absolute error values falling below 0.005. Near the domain boundaries, where the baseline model showed greater departures from the genuine solution, the improvement is most apparent. These findings show that Gradient Boosting may iteratively improve the ELM's predictions, which makes it a feasible method for raising

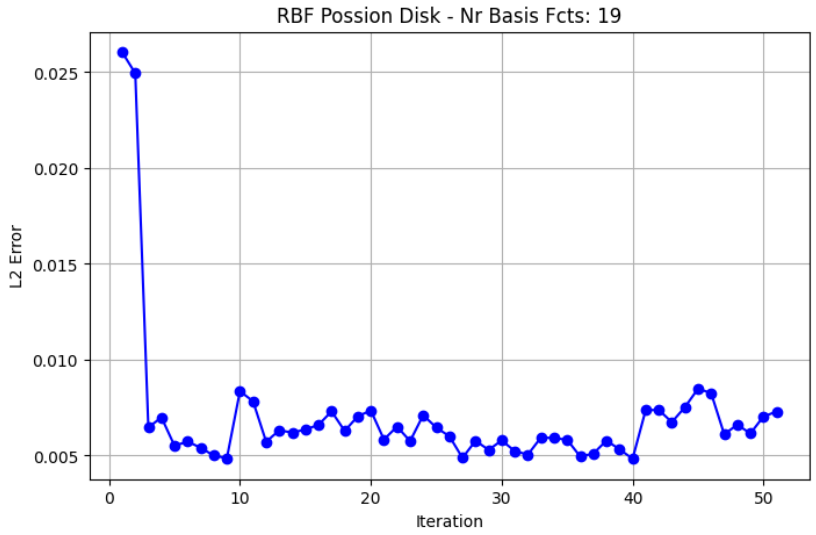

Figure 5.1: Evolution of the L_2-error over 50 Gradient Boosting iterations.

the precision of PDE solvers based on Neural Networks. Despite its success, the boosting process has revealed computational and practical challenges. One of the most important bottlenecks despite the residuals were the combined models, especially after more than 10-20 iterations. This step was computationally expensive, scaling the complexity of the model.

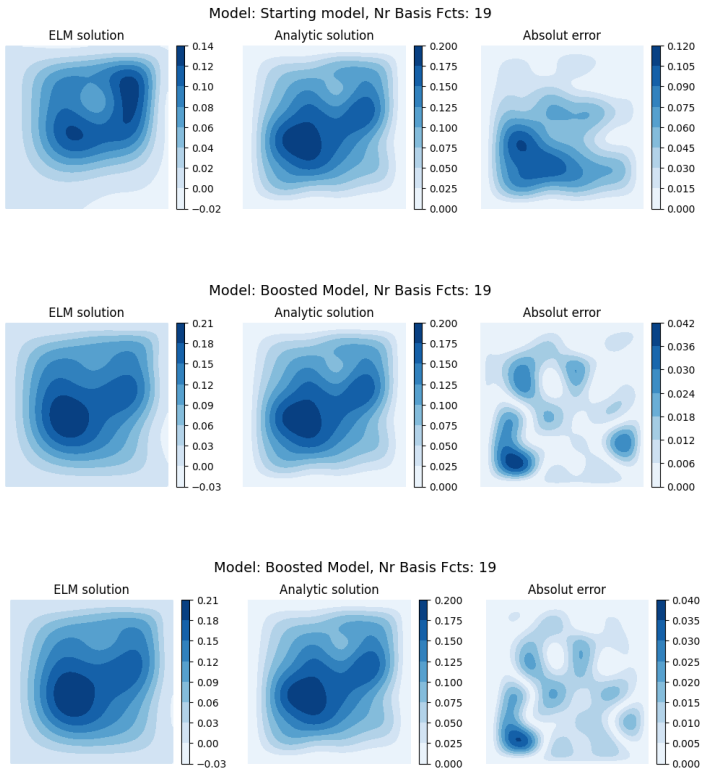

Figure 5.2: Solution after 50 boosting iterations.

5.3 Outlook on Adaptive Boosting Strategies and Incremental Methods

Traditional boosting methods, particularly gradient boosting, often introduce computational overhead and may not optimally exploit the structure of the problem. To enhance efficiency, stability, and adaptability, future research should focus on more refined boosting approaches, including adaptive learning rate adjustments, incremental model updates, and hybrid boosting methods.

A critical limitation in standard boosting techniques is the use of a fixed learning rate (α), which can lead to oscillatory behavior or slow convergence. *Dynamic learning rate adjustment* optimizes α by refining it based on error gradients, ensuring aggressive corrections in early iterations and fine-tuned adjustments in later stages. By implementing this, the model would achieve faster convergence with fewer iterations, reducing computational overhead while improving stability. Instead of requiring manual fine-tuning, this method would allow the learning rate to adapt dynamically, making the approach more robust across different PDE problems.

Another approach are Convex Incremental Extreme Learning Machines (CI-ELMs), which offer an alternative to traditional boosting by introducing iterative model refinements without retraining entire networks at each step. Instead of training multiple separate models, CI-ELMs incrementally adjust weight parameters based on residual gradients, significantly reducing computational overhead.

Implementing CI-ELMs in the PIELM framework would provide several improvements. First, it would drastically reduce memory consumption and computation time, as only incremental updates to the model are performed instead of training new models at each iteration. Second, faster convergence would be achieved, as the model does not need to be reset at each boosting step, allowing it to build on previous iterations more effectively [21].

By integrating these strategies, PIELMs could achieve good performance in large-scale simulations, making them more practical for large scientific applications.

6 Conclusion

This thesis explored the development of Physics-Informed Extreme Learning Machines for solving the Poisson equation, addressing the potential challenges of traditional numerical methods such as high computational costs, ill-conditioning and solving for higher dimensions. By integrating physics constraints directly into ELM architectures, PIELMs provide an efficient alternative to iterative optimization approaches. The study systematically analyzed the impact of different basis functions, including Radial Basis Functions (RBFs), B-Splines, Chebyshev polynomials, and Fourier series, on the accuracy and stability of PIELM solutions. The results demonstrated that the choice of basis functions significantly influences the condition number and convergence properties of the method.

Among the key findings, RBFs with Poisson Disk Sampling proved to be highly effective in reducing numerical instability while maintaining good approximation accuracy. Chebyshev polynomials, Fourier Series and B-Splines provided well-conditioned representations. Additionally, the thesis investigated the role of domain sampling methods, showing that Poisson Disk Sampling improves basis function distribution and mitigates ill-conditioning. The study also explored boosting techniques to refine solution accuracy, analyzing their potential benefits, computational costs while highlighting the need for balancing computational efficiency.

The insights gained from this work can be extended to more complex applications in magnetostatics, micromagnetism, and fluid dynamics, where fast and stable solvers are crucial for large-scale simulations. A particularly promising future direction is the integration of PIELMs with the splitting ansatz proposed by Garcia-Cervera and Roma, which decomposes PDE problems into two subproblems that are solved within the magnetic domain. The stray field problem in magnetostatics should also be discussed, in particular how PIELMs could be of advantage, unfortunately the realization was not possible due to time constraints and is left for future research.

In addition to these approaches, the application of Convex Incremental Extreme Learning Machines (CI-ELMs) [21] could offer improvements in stability and adaptive learning within the PIELM framework. Since CI-ELMs introduce incremental updates with convex optimization, they could help mitigate conditioning issues in high-function-count Chebyshev polynomials or RBFs, making PIELMs more scalable and robust.

Further improvements to PIELMs could involve preconditioning techniques to mitigate ill-conditioning, particularly for Chebyshev polynomials and RBFs, which showed instability at higher function counts. Incorporating adaptive sampling strategies tailored to PIELMs could further enhance stability and accuracy, ensuring that sampling density aligns with regions of high solution variation. Moreover, integrating boosting techniques into PIELMs, such as adaptive boosting or gradient boosting for basis function refinement, could improve convergence and numerical robustness.

Recent advancements in deep operator learning present additional opportunities for enhancing PIELMs. DeepONets could potentially be used to generalize across different Poisson equation settings by learning function-to-function mappings that adapt to different problem configurations. Fourier Neural Operators (FNOs) [30], which leverage spectral representations for efficient PDE solving, could be explored as a tool for improving PIELM performance,

particularly for problems with inherent periodicity or large-scale interactions. A promising direction would be to use FNOs to pre-train the feature space of PIELMs, allowing for more structured weight initialization and better spectral representations. Similarly, hybrid architectures could be investigated, where DeepONets optimize the selection of basis functions dynamically, leading to adaptive function spaces tailored to problem-specific constraints.

While this study provides a detailed evaluation of basis function performance within the PIELM framework, future research could explore adaptive hybrid methods, convex incremental learning strategies, deep-learning-assisted basis selection, and scalable approaches for three-dimensional (3D) and time-dependent PDEs, which were not yet covered hence not enough time could be devoted to it. The scalability of PIELMs in 3D micromagnetic simulations remains an important open question, as higher-dimensional problems introduce additional computational challenges that require further numerical optimization. Extending PIELMs to solve full micromagnetic models, including the Landau-Lifshitz-Gilbert (LLG) equation, could provide a more complete picture of how machine learning techniques can improve computational efficiency in large-scale physics simulations. Moreover, applying PIELMs to time-dependent PDEs could allow for more efficient modeling of magnetization dynamics in magnetic materials.

This research establishes a strong foundation for advancing physics-informed computational methods, demonstrating the potential of machine learning in solving Partial Differential Equations with greater efficiency, stability, and scalability. By leveraging insights gained from basis function comparisons, this work contributes to the broader effort of bridging numerical methods with physics-informed machine learning, opening new possibilities for the next generation of PDE solvers in micromagnetism, computational physics, and beyond.

Bibliography

[1] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Books on Mathematics. Dover Publications, Mineola, NY, second edition, 2001. 34

[2] Paul Bratley and Bennett L. Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Trans. Math. Softw.*, 14(1):88–100, March 1988. 18, 19

[3] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 3rd edition, 2008. 1

[4] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *Proceedings of Fast Poisson Disk Sampling in Arbitrary Dimensions*. University of British Columbia, 2007. 20

[5] Steven L. Brunton and J. Nathan Kutz. Machine learning for partial differential equations. *arXiv preprint arXiv:2303.17078*, 2023. 6, 18

[6] Martin D Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003. 17

[7] Carl de Boor. *A Practical Guide to Splines*. Springer, 2001. 25

[8] V. Dwivedi and B. Srinivasan. Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations. *Applied Numerical Mathematics*, No. (Volume, if known):No. (Page numbers, if known), 2021. 2

[9] Lawrence C Evans. *Partial Differential Equations*. American Mathematical Society, 2010. 1, 4

[10] L. Exl. Lecture notes: Numerical mathematics 2. Partial Differential Equations and Numerical Techniques, 2024. 1

[11] L. Exl and S. Schaffer. Angewandtes maschinelles lernen. Lecture Notes for Applied Machine Learning, Winter Semester 2024, 2024. 42, 43

[12] Stanley J. Farlow. *Partial Differential Equations for Scientists and Engineers*. Dover Publications, 1993. 1, 4

[13] Josef Fidler and Thomas Schrefl. Micromagnetic modelling—the current state of the art. *Journal of Physics D: Applied Physics*, 33(15):R135–R156, 2000. 6

[14] Gene H Golub and Charles F Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013. 1

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 2, 7, 9

[16] H. Guo, J. Wang, W. Ao, and Y. He. Sgb-elm: An advanced stochastic gradient boosting-based ensemble scheme for extreme learning machine. *Journal of Machine Learning Research*, 2020. 2, 42

[17] Jiequn Han, Arnulf Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018. 2, 7

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. 7, 9

[19] G.-B. Huang and C. K. Siew. Extreme learning machine: Rbf network case. In *Proceedings of the 2004 8th Control, Automation, Robotics and Vision Conference (ICARCV)*, volume 2, pages 1024–1028. IEEE Xplore, 2005. 16, 42

[20] Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003. 22

[21] Guang-Bin Huang and Lei Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16):3056–3062, 2007. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005). 3, 45, 46

[22] Guang-Bin Huang, Di Wang, and Yuan Lan. Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics*, 2:107–122, 2011. 2, 9

[23] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. 2, 9, 22

[24] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. 6

[25] Thomas J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Courier Corporation, 2012. 5

[26] John David Jackson. *Classical Electrodynamics*. Wiley, 3rd edition, 1999. 1, 4

[27] George E. Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440, 2021. 2, 8

[28] C. Kirisits. Lecture notes: Numerical mathematics 1. Focus on Partial Differential Equations, 2024. 4

[29] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, 2007. 5

[30] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. 3, 46

[31] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. 3, 6, 8

[32] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991. 29

[33] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. 2, 6, 7, 8

[34] Martin Roberts. The unreasonable effectiveness of quasirandom sequences. https://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/, 2018. Accessed: 2025-01-21. 19

[35] V. Rvachev, T. Sheiko, V. Shapiro, and Igor Tsukanov. On completeness of rfm solution structures. *Computational Mechanics*, 25:305–317, 03 2000. 14

[36] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003. 5

[37] S. Schaffer. *Physics Informed Machine Learning in the Field of Micromagnetism*. PhD thesis, University of Vienna, 2024. Dissertation, in partial fulfillment of the requirements for the degree of Doktor der Naturwissenschaften (Dr. rer. Nat). 1, 2, 13, 14, 15, 16, 22

[38] S. Schaffer. Magpi - a collection of useful algorithms and methods for the training of physics informed neural networks (pinns). https://github.com/schaffer9/MagPI, 2025. Accessed: 2025-03-17. 24

[39] S. Schaffer and L. Exl. Constraint free physics-informed machine learning for micromagnetic energy minimization. *Computer Physics Communications*, 300:109202, 2024. 1, 2

[40] V. Shapiro. Theory of r-functions and applications: A primer. Technical report, Cornell University, 1991. 14

[41] Vadim Shapiro. Semi-analytic geometry with r-functions. *Acta Numerica*, 16:239–303, 2007. 14

[42] Daniel S. Soper. Using an opportunity matrix to select centers for rbf neural networks. *Algorithms*, 16(10), 2023. 18

[43] Walter A. Strauss. *Partial Differential Equations: An Introduction*. Wiley, 2nd edition, 2008. 4

[44] Lloyd N Trefethen and David Bau III. *Numerical Linear Algebra*. SIAM, 1997. 1, 5

[45] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, and Jian Zhong Zhu. *The Finite Element Method for Solid and Structural Mechanics*. Butterworth-Heinemann, 2005. 5

List of Figures

List of Tables