Journal of Magnetism and Magnetic Materials 548 (2022) 168951

Contents lists available at ScienceDirect

Journal of Magnetism and Magnetic Materials

journal homepage: www.elsevier.com/locate/jmmm

Research articles

Magnetostatics and micromagnetics with physics informed neural networks

Alexander Kovacs [a,b], Lukas Exl [c,d,e], Alexander Kornell [a,b], Johann Fischbacher [a,b], Markus Hovorka [a,b], Markus Gusenbauer [a,b], Leoni Breth [b], Harald Oezelt [b], Dirk Praetorius [f], Dieter Suess [g,e], Thomas Schrefl [a,b,e,*]

[a] Christian Doppler Laboratory for Magnet design through physics informed machine learning, Viktor Kaplan-Straße 2E, 2700 Wiener Neustadt, Austria
[b] Department for Integrated Sensor Systems, Danube University Krems, Viktor Kaplan-Straße 2E, 2700 Wiener Neustadt, Austria
[c] Department of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
[d] Wolfgang Pauli Institute, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
[e] Research Platform MMM Mathematics-Magnetism-Materials, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
[f] Institute of Analysis and Scientific Computing, TU Wien, Wiedner Hauptstraße 8-10, 1040 Vienna, Austria
[g] Physics of Functional Materials, University of Vienna, Währinger Straße 17, 1090 Vienna, Austria

ARTICLE INFO

Keywords:
Magnetostatics
Neural network
Ritz method
Inverse problems

ABSTRACT

Partial differential equations and variational problems can be solved with physics informed neural networks (PINNs). The unknown field is approximated with neural networks. Minimizing the residuals of the static Maxwell equation at collocation points or the magnetostatic energy, the weights of the neural network are adjusted so that the neural network solution approximates the magnetic vector potential. This way, the magnetic flux density for a given magnetization distribution can be estimated. With the magnetization as an additional unknown, inverse magnetostatic problems can be solved. Augmenting the magnetostatic energy with additional energy terms, micromagnetic problems can be solved. We demonstrate the use of physics informed neural networks for solving magnetostatic problems, computing the magnetization for inverse problems, and calculating the demagnetization curves for two-dimensional geometries.

1. Introduction

The development and optimization of magnetic devices involves, among other tasks, the calculation of magnetic flux, the solution of inverse magnetostatic problems, and the calculation of hysteresis loops of magnetic materials. Examples on the macroscopic scale are magnets in accelerators and electron storage rings [1], magnetic write heads in magnetic recording [2], and permanent magnet systems with a predefined stray field for example for sensor applications [3]. In many applications, a magnetostatic field with predefined properties is sought: Accelerator magnets should produce fields that are either uniform or vary linearly in space. Magnetic recording heads that create fields with a high field gradient are essential to achieve high storage densities. In order to reach fields with certain properties, either the shape of the field source or the magnetization distribution within the field generating magnet [3] or both can be optimized. For shape optimization, on/off methods [4,5], in which space points are either magnetic or non-magnetic, or parameterized geometries [6] have been used. Inverse problems, in which the optimal distribution of the magnetization is to be found, are efficiently solved with the adjoint method [7].

On a microscopic scale the computation of the magnetization inside a magnetic material is of importance. Solving for the magnetization as function of the external field gives the hysteresis loop. The magnetization distribution is the solution of Brown's micromagnetic equation [8]. Micromagnetics addresses the interplay between the local chemical composition, the microstructure of the material, and the hysteresis properties [9,10]. Local material properties are reflected by the coefficients of the partial differential equation (PDE).

Traditionally, the numerical solution of (inverse) magnetostatic and micromagnetic problems relies on the finite difference or finite element discretization of the underlying partial differential equations. For the fast estimation of magnetostatic fields in motors [11] or the magnetic response of magnetic sensor elements, neural networks [11,12] or kernel methods [13] have been applied. In order to train the machine learning models, conventional numerical solvers are used to generate the training data by varying geometry, external loads, or time. This makes the numerical solution of the partial differential equations a pre-processing step. However, once the machine learning model is trained, the magnetic field or the magnetization can be quickly estimated. Such

* Corresponding author at: Department for Integrated Sensor Systems, Danube University Krems, Viktor Kaplan-Straße 2E, 2700 Wiener Neustadt, Austria.
E-mail address: thomas.schrefl@donau-uni.ac.at (T. Schrefl).

https://doi.org/10.1016/j.jmmm.2021.168951
Received 7 June 2021; Received in revised form 5 November 2021; Accepted 10 December 2021
Available online 1 January 2022

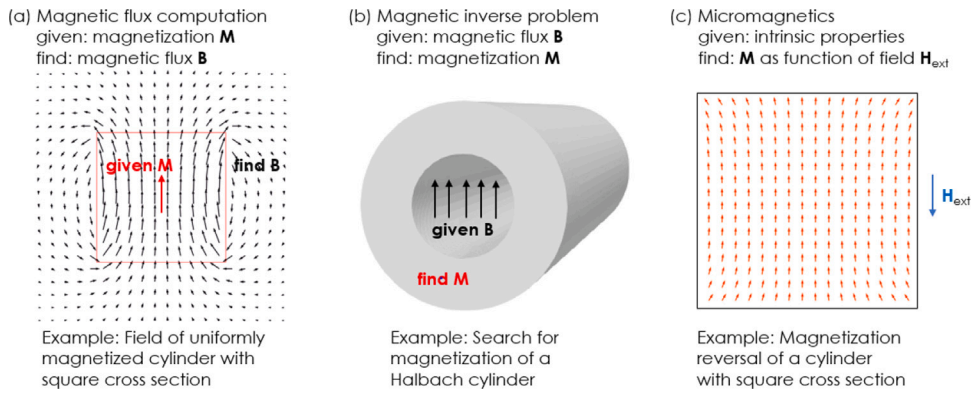

Fig. 1. Applications for physics informed neural networks addressed in this work. (a) Magnetostatic field computation: Estimation of the magnetic flux of a permanent magnet. (b) Magnetostatic inverse problem: Estimation of the magnetization distribution for a Halbach cylinder. (c) Micromagnetics: Computation of the hysteresis loop of a hard magnetic particle. To demonstrate the use of physics informed neural networks in magnetostatics and micromagentics for two-dimensional problems. The magnets are infinitely extended in one direction. This is schematically shown in (b). The magnets in (a) and (b) are infinitely extended in the direction normal to the drawing plane.

models are useful for interactive design, optimization, or real time applications.

An alternative way for solving partial differential equations numerically are physics informed neural networks [14]. The loss function of physics informed neural networks, which is minimized during training, is directly computed from the governing partial differential equation. The loss function is either formed by the residuals at collocation points [15], the weighted residuals obtained by the Galerkin method [16], or the energy functional of an Euler–Lagrange differential equation [17]. For training a physics informed neural network, there is no need to generate training data in advance. The input data for physics informed neural networks are points sampled in the problem domain. Generally, training of machine learning models can be supervised or unsupervised. Learning a function giving a set of example input–output pairs is called supervised learning. In supervised learning the loss function is usually the mean squared error between the output values of the network and the true values from the training pairs. A training pair is an example for a map between input features and (an) output label(s). During training, the unknown weights of the network are modified such that the loss function is minimized. In physics informed neural networks the loss function does not depend on precomputed labels that are usually given by the computed solution of the partial differential equation. The loss function in physics informed neural networks is based on the residuals of the partial differential equation or an energy functional. Therefore, training can be achieved in an unsupervised way. The training process has some similarities to the numerical solution of a partial differential equation with the finite element method. Within the framework of the finite element method, the unknown solution is expressed with a set of basis function and the expansion coefficients are found by minimizing residuals or an energy functional. In physics informed neural networks, the unknown solution is approximated by a neural network. Similar as in the finite element method, the trainable weights of the neural network are found by minimizing residuals or an energy functional. The training data of physics informed neural networks are the Cartesian coordinates of quasi-randomly sampled points in the problem domain.

The loss function of physics informed neural networks can be augmented with the distance between the approximated solution of the partial differential equation and desired values of the solution at given points in space. Then, one or more coefficients of the partial differential equation can be included as unknowns during training. Solving inverse problems with physics informed neural networks may lead to a significant speed up as compared to conventional methods [18].

In this work, we demonstrate the use of physics informed neural networks for inverse magnetostatic and micromagnetic problems. We first show how physics informed neural networks can be used to solve the forward magnetostatic problem. We use a set of dense neural networks [17] to approximate the unknown magnetic vector potential. During training, the residuals related to the partial differential equation and boundary and interface conditions are minimized by adjusting the weights of the network. For solving inverse problems, a second set of neural networks approximates the unknown magnetization. An additional condition is added to the sum of residuals that penalizes the difference between the desired field and the current numerical estimate.

Alternatively, we can compute the magnetic flux density created from a given magnetization distribution by minimizing Brown's upper limit for the magnetostatic energy [8]. During training, the magnetostatic energy is minimized by adjusting the weights of the network. This approach is similar to the Ritz method for computing the magnetic field: However, instead of finite element basis functions [19], we use dense neural networks [17] to approximate the unknowns. Again we can introduce the magnetization as additional unknown and approximate it with a set of neural networks. We can minimize the total micromagnetic energy, which is the sum of the magnetostatic energy, the ferromagnetic exchange energy, the magneto-crystalline anisotropy energy, and the exchange energy by simultaneously adjusting the weights in the neural networks for the magnetic vector potential and the magnetization. This joint minimization of the energy with respect to the magnetic vector potential and the magnetization for the numerical solution of micromagnetic problems was suggested by Asselin and Thiele [20] and applied in finite element micromagnetics for soft magnetic elements [21] and permanent magnets [22].

This work is a proof of concept for the successful application of physics informed neural networks for the solution of magnetostatic and micromagnetic problems. The solution of electric and magnetic field problems is easier and requires less computation time if three-dimensional problems are approximated in two dimensions. In two dimensions the field formulations are simpler, and the number of partial differential equations needed to be solved simultaneously is smaller. Two-dimensional approximations in which all quantities are assumed to be constant in the direction normal to the plane of computation were quite popular in magnetic field problems [23] and micromagnetics [21,24]. Similarly, the simplicity of two-dimensional field equations makes it easier to apply a physics informed neural networks in computational magnetism. However, it should be straightforward to extend the methodology to three dimensions. This would increase the number of neural networks to be trained for solving a field problem and increase training time accordingly.

Fig. 1 shows the problems addressed with physics informed neural networks within this work. Whenever possible we will focus on simple problems for which analytical solutions are known. In a magnetostatic problem we compute the magnetic flux density for a given magnetization distribution. As shown in Fig. 1a, we will compute the magnetic

flux density of a uniformly magnetized particle [24] with physics informed neural networks. In magnetostatic inverse problems we search for the magnetization when the magnetic flux density is given. As shown in Fig. 1a, we will compute the magnetization distribution in the ring of a Halbach cylinder [25]. In micromagnetics we search for the magnetization distribution as a function of the applied field as shown in Fig. 1c. Please note that for this problem we also have to take care of hysteresis: The magnetic states depend on the history of the applied field. Here we compute the coercive field of a $Nd_2Fe_{14}B$ particle [22]. We will restrict ourselves to two-dimensional problems. In the past, two-dimensional micromagnetic simulations of permanent magnets were found to give reasonable lower bounds for the coercive field [22]. The comparison of experimental data with two-dimensional micromagnetic results showed excellent agreement for the remanent magnetization and the coercive field [26].

Like any other machine learning model, physics informed neural networks need to be trained. Training involves the solution of a high-dimensional non-convex optimization problem. The solution of single forward problems is much faster with conventional numerical methods [27] such as the finite difference or the finite element method. For example, the total time to solution, which includes training and field evaluation of the magnetostatic field problem with the deep Ritz method (see Section 4.1) was 18 min on a Tesla V100-PCIE-32 GB card. However, once trained, the physics informed machine neural network can give the solution quickly. Therefore, physics informed neural networks are advantageous for solving parametric partial differential equations. Once trained, the neural network gives solutions for an entire class of partial differential equations [28,29]. Therefore, physics informed neural networks are several orders of magnitudes faster for design optimization problems. Hennigh and co-workers [18] report significant time savings for heat sink optimization as compared to a commercial fluid dynamics solver by using a parameterized geometry and physics informed neural networks. Karniadakis and co-workers [27] give an overview of application areas and discuss the merits of physics informed neural networks. Physics informed neural networks are very efficient for the solution of inverse problems. In this paper we demonstrate the use of physics informed neural networks for the solution of partial differential equation in applied magnetism. This is a necessary first step towards future use of physics informed neural networks for optimizing magnetic materials and devices.

The paper is organized as follows. We will first introduce the governing partial differential equations of magnetostatics and the micromagnetic energy functional. Then we will show how the solutions can be approximated with neural networks and we will define the loss functions associated with each problem. Finally, we will discuss the numerical results.

2. Micromagnetic background

2.1. Two-dimensional magnetostatics

We are interested in computing the magnetic flux density, **B**, or the magnetic field, **H**, for a given magnetization distribution, **M**. In magnetostatics we have no time dependent quantities. In the presence of a stationary current Maxwell's equations reduce to [23]

$$\nabla \times \mathbf{H} = \mathbf{j}, \tag{1}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{2}$$

Here **j** is the current density. The current density fulfills $\nabla \cdot \mathbf{j} = 0$ which expresses the conservation of electric charge. On a macroscopic length scale the relation between the magnetic induction and the magnetic field is expressed by

$$\mathbf{B} = \mu \mathbf{H}, \tag{3}$$

where μ is the permeability of the material. Eq. (3) is used in magnetostatic field solvers [23] for the design of magnetic circuits. In these simulations, the permeability describes the response of the material to the magnetic field. The influence of the material can also be expressed by its magnetization distribution $\mathbf{M}(\mathbf{x})$. Then we use

$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}) \tag{4}$$

instead of (3). Here μ_0 is the permeability of vacuum. For now, let us assume $\mathbf{j} = 0$. Taking the curl of (4) and plugging into (1) gives

$$\nabla \times \mathbf{B} = \mu_0 \nabla \times \mathbf{M}. \tag{5}$$

In numerical calculations, the constraint (2) that **B** is solenoidal can be fulfilled by introducing a magnetic vector potential $\nabla \times \mathbf{A} = \mathbf{B}$. Thus, we arrive at

$$\nabla \times (\nabla \times \mathbf{A}) = \mu_0 \nabla \times \mathbf{M}. \tag{6}$$

We now assume that the magnetic sources are infinitely extended in direction x_3 and that the magnetization **M** is translationally invariant in x_3. Then the x_3 component of the magnetic induction is constant. The problem reduces to two dimensions [20] with $\mathbf{M} = \mathbf{M}(x_1, x_2)$, $\mathbf{B} = \mathbf{B}(x_1, x_2)$, and $A = A_{x_3}(x_1, x_2)$. For simplicity, we simply write A for the x_3-component of the magnetic vector potential. The components of the magnetic flux density are [25]

$$B_{x_1} = \frac{\partial A}{\partial x_2}, \; B_{x_2} = -\frac{\partial A}{\partial x_1}. \tag{7}$$

Using the vector identity $\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2 \mathbf{A}$ and translational invariance in x_3 direction, we rewrite (6) as

$$\frac{\partial^2 A}{\partial x_1^2} + \frac{\partial^2 A}{\partial x_2^2} = \mu_0 \left(\frac{\partial M_{x_1}}{\partial x_2} - \frac{\partial M_{x_2}}{\partial x_1} \right). \tag{8}$$

Eq. (8) has to be fulfilled inside the magnetic material where $|\mathbf{M}| > 0$. Outside of the magnet, the magnetic vector potential solves the Laplace equation

$$\frac{\partial^2 A}{\partial x_1^2} + \frac{\partial^2 A}{\partial x_2^2} = 0. \tag{9}$$

At the surface of the magnetic body with normal vector **n**, the normal component of the magnetic flux density and the tangential component of the magnetic field are continuous

$$\left(\mathbf{B}^{(\text{in})} - \mathbf{B}^{(\text{out})}\right) \cdot \mathbf{n} = 0, \tag{10}$$

$$\left(\mathbf{H}^{(\text{in})} - \mathbf{H}^{(\text{out})}\right) \times \mathbf{n} = 0. \tag{11}$$

In terms of the magnetic vector potential and the magnetization, the interface conditions are

$$\left(\frac{\partial A^{(\text{in})}}{\partial x_2} - \frac{\partial A^{(\text{out})}}{\partial x_2} \right) n_{x_1} + \left(\frac{\partial A^{(\text{out})}}{\partial x_1} - \frac{\partial A^{(\text{in})}}{\partial x_1} \right) n_{x_2} = 0, \tag{12}$$

$$\left(\frac{\partial A^{(\text{in})}}{\partial x_2} - \frac{\partial A^{(\text{out})}}{\partial x_2} - \mu_0 M_{x_1} \right) n_{x_2} - \left(\frac{\partial A^{(\text{out})}}{\partial x_1} - \frac{\partial A^{(\text{in})}}{\partial x_1} - \mu_0 M_{x_2} \right) n_{x_1} = 0. \tag{13}$$

At infinity the magnetic flux density approaches zero.

Alternatively, we can minimize a sharp upper bound of the magnetostatic energy. Brown [30] suggested a functional to be used as upper bound for the magnetostatic energy. The magnetostatic energy E_m due to the magnetization $\mathbf{M}(\mathbf{x})$ can be bounded by a functional [22]

$$E_\text{m} \le W_\text{m}(\mathbf{B}') = \frac{1}{2\mu_0} \int \left(\mathbf{B}' - \mu_0 \mathbf{M}\right)^2 \mathrm{d}^3 x. \tag{14}$$

The functional $W_\text{m}(\mathbf{B}')$ if minimized subject to the constraint $\nabla \cdot \mathbf{B}' = 0$, makes $\mathbf{B}'$ equal to the magnetic induction **B** created by the magnetization **M**. Again, we can introduce a magnetic vector potential to make $\mathbf{B}'$ solenoidal. The Euler–Lagrange equation of (14) with respect to the magnetic vector potential gives the partial differential Eq. (6) [20].

Assuming translational symmetry in x_3 direction, we obtain

$$W_\text{m}(A') = \frac{1}{2\mu_0} \int \left(\left(\frac{\partial A'}{\partial x_1} \right)^2 + \left(\frac{\partial A'}{\partial x_2} \right)^2 \right.$$

$$+2\mu_0\left(M_{x_2}\frac{\partial A'}{\partial x_1}-M_{x_1}\frac{\partial A'}{\partial x_2}\right)+\mu_0^2\left(M_{x_1}^2+M_{x_2}^2\right)\Big)\,\mathrm{d}^2x. \quad (15)$$

If minimized with respect to A', the functional (15) gives the magnetostatic energy. Please note that the last term in (15) adds a constant offset to the magnetostatic energy and thus may be dropped.

The integral in (15) is over the entire space. Thus, when evaluating (15), we have to integrate over the magnet, where $M_{x_1}\neq 0$ or $M_{x_2}\neq 0$, and over a large region outside the magnetic material. For integration we simply truncate the infinite region outside the magnet. To keep the truncation error small, the distance of the outer boundary to the center of the magnet should be at least five times the distance of the center of the magnet to its most remote outer surface [31].

2.2. Two-dimensional micromagnetics

In micromagnetics, we want to compute the local distribution of the magnetization as function of the magnetic field. This is the response of the system to a(n external) field. We assume translational symmetry in a direction perpendicular to the plane containing anisotropy axis and the external field. In other words, we consider a particle which is infinitely extended in the direction perpendicular to the plane which contains the anisotropy axis [32]. If the magnetization is initially parallel to the anisotropy axis it remains in the plane upon energy minimization. For a given value of the external field $\mathbf{H}_{\mathrm{ext}}$, the magnetization distribution $\mathbf{M}=\mathbf{M}(x_1,x_2)$ can be derived from the minimization of the total Gibbs free energy. During minimization, the constraint $|\mathbf{M}|=M_{\mathrm{s}}$ has to be fulfilled. The spontaneous magnetization M_{s} of a material depends on temperature, but is independent of the external field [8]. The hysteresis loop follows from the path formed by subsequently following local minima in an energy landscape progressively changed by a varying external field [33].

The Gibbs free energy E is the sum of the magnetostatic energy E_{m}, the Zeeman energy of the magnetization in an external field, the magneto-crystalline anisotropy energy, and the ferromagnetic exchange energy [8]. For an efficient numerical scheme we follow Asselin and Thiele [20] and replace E_{m} with W_{m} as given by (15). We define an upper bound of the total energy

$$W(\mathbf{M},A')=W_{\mathrm{m}}(\mathbf{M},A')+\int_{V^{(\mathrm{in})}}\Big\{-\mu_0\mathbf{M}\cdot\mathbf{H}_{\mathrm{ext}}+K_1\sin^2\alpha+K_2\sin^4\alpha+\frac{C}{M_{\mathrm{s}}^2}\left[\left(\nabla M_{x_1}\right)^2+\left(\nabla M_{x_2}\right)^2\right]\Big\}\,\mathrm{d}^2x. \quad (16)$$

The local minima of the auxiliary functional W are in one-to-one correspondence with those of the total Gibbs free energy E. Here K_1, K_2 are the anisotropy constants, α is the angle between the magnetization $\mathbf{M}$ and the anisotropy direction, and C is the exchange constant. The second term on the right hand side of (16) is over the volume of the magnet $V^{(\mathrm{in})}$ where $|\mathbf{M}|>0$. The right hand side of (16) contains $W_m(\mathbf{M},A')$, which is an upper bound for the magnetostatic energy. Minimization of the right hand side of (16) with respect to A' makes $W_m(\mathbf{M},A')$ equal to the magnetostatic energy $E_{\mathrm{m}}=E_{\mathrm{m}}(\mathbf{M})$.

3. Physics informed neural networks

3.1. Collocation based magnetostatics

For creating the neural network approximation, we follow the approach of Niakia and co-workers [34] and introduce distinct neural networks for the different regions of the problem domain. The two neural networks approximate the vector potentials inside the magnetic domain, $A^{(\mathrm{in})}$, and outside the magnetic domain, $A^{(\mathrm{out})}$, respectively. The interface conditions (12) and (13) are incorporated into the loss function. The inputs of the neural networks are points in the two-dimensional problem domain (x_1,x_2), the output of the neural network is the approximation of the magnetic vector potential:

$$A^{(\mathrm{in})}_{\mathrm{approx}}=\mathcal{N}_{A^{(\mathrm{in})}}(x_1,x_2,\mathbf{w}_{A^{(\mathrm{in})}}), \quad (17)$$

$$A^{(\mathrm{out})}_{\mathrm{approx}}=\mathcal{N}_{A^{(\mathrm{out})}}(x_1,x_2,\mathbf{w}_{A^{(\mathrm{out})}}). \quad (18)$$

The vectors $\mathbf{w}_{A^{(\mathrm{in})}}$ and $\mathbf{w}_{A^{(\mathrm{out})}}$ represent the weights and biases of each network. The location of a point in the problem domain is given by the vector $\mathbf{x}=(x_1,x_2)$. The weights and biases are the learnable parameters of the networks which are determined during training of the networks by minimizing the sum of the squared residuals at collocation points. During training of the neural networks (17) and (18) the weights and biases are adjusted so that $A^{(\mathrm{in})}_{\mathrm{approx}}$ is an approximate solution of Eq. (8), $A^{(\mathrm{out})}_{\mathrm{approx}}$ is an approximate solution of Eq. (9), and both fulfill the interface conditions (12) and (13). The magnetic flux should decay to zero as $|\mathbf{x}|$ approaches infinity. To account for this condition we expand the problem domain up to a certain distance outside the magnetic body and force $\mathbf{B}=0$ at the boundary of the truncated problem domain. The two networks are trained simultaneously. The loss function for the joint training of the two networks is the following sum

$$L_{\mathrm{collocation}}=L_{A^{(\mathrm{in})}}+L_{A^{(\mathrm{out})}}+L_{\mathbf{B}^{(\mathrm{out})}}+L_{B_{\mathrm{n}}}+L_{H_{\mathrm{t}}}. \quad (19)$$

We define the following indicator functions or binary masks

$$in(\mathbf{x})=\begin{cases}1 & \text{for } \mathbf{x} \text{ inside } V^{\mathrm{in}}\\ 0 & \text{otherwise}\end{cases} \quad (20)$$

$$bnd(\mathbf{x})=\begin{cases}1 & \text{for } \mathbf{x} \text{ close to the surface of } V^{\mathrm{in}}\\ 0 & \text{otherwise}\end{cases} \quad (21)$$

$$inf(\mathbf{x})=\begin{cases}1 & \text{for } \mathbf{x} \text{ close to the outer boundary of the problem domain}\\ 0 & \text{otherwise}\end{cases} \quad (22)$$

in order to track the location of the collocation points $\mathbf{x}_i$.

The loss $L_{A^{(\mathrm{in})}}$ is the mean squared sum of the residuals of (8)

$$L_{A^{(\mathrm{in})}}=\frac{1}{n}\sum_{i=1}^{n}in(\mathbf{x}_i)\left(\left.\frac{\partial^2A^{(\mathrm{in})}_{\mathrm{approx}}(\mathbf{x})}{\partial x_1^2}\right|_{\mathbf{x}=\mathbf{x}_i}+\left.\frac{\partial^2A^{(\mathrm{in})}_{\mathrm{approx}}(\mathbf{x})}{\partial x_2^2}\right|_{\mathbf{x}=\mathbf{x}_i}-\mu_0\left(\left.\frac{\partial M_{x_1}(\mathbf{x})}{\partial x_2}\right|_{\mathbf{x}=\mathbf{x}_i}-\left.\frac{\partial M_{x_2}(\mathbf{x})}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_i}\right)\right)^2. \quad (23)$$

Terms denoted with the subscript 'approx' such as $A^{(\mathrm{in})}_{\mathrm{approx}}(\mathbf{x})$ are the neural network approximations of the respective physical quantities. The units of the hidden layers in a neural network typically apply a non-linear activation function after the linear combination of the inputs. Today many neural networks use rectified linear units with the activation function $g(z)=\max\{0,z\}$. Rectification was found to increase the performance of feed-forward neural networks [35]. The activation function $g(z)$ is non-differentiable at $z=0$. For this work, we decided to avoid activation functions with non-differentiable points and use the hyperbolic tangent as activation function for all hidden units. With this choice of activation function, the neural network approximation is infinitely differentiable. The loss $L_{A^{(\mathrm{out})}}$ is the mean squared sum of the residuals of (9)

$$L_{A^{(\mathrm{out})}}=\frac{1}{n}\sum_{i=1}^{n}\left(1-in(\mathbf{x}_i)\right)\left(\left.\frac{\partial^2A^{(\mathrm{out})}_{\mathrm{approx}}(\mathbf{x})}{\partial x_1^2}\right|_{\mathbf{x}=\mathbf{x}_i}+\left.\frac{\partial^2A^{(\mathrm{out})}_{\mathrm{approx}}(\mathbf{x})}{\partial x_2^2}\right|_{\mathbf{x}=\mathbf{x}_i}\right)^2. \quad (24)$$

Plugging $A^{(\mathrm{in})}_{\mathrm{approx}}$ and $A^{(\mathrm{out})}_{\mathrm{approx}}$ into Eq. (7), we get the neural network approximations of the magnetic flux density inside $\mathbf{B}^{(\mathrm{in})}_{\mathrm{approx}}$ and outside the magnet $\mathbf{B}^{(\mathrm{out})}_{\mathrm{approx}}$. By definition of the magnetostatic boundary value problem (see Section 2.1) the magnetic flux density must vanish at infinity. To account for this boundary condition, we introduce an additional loss function. Upon minimization, the loss $L_{\mathbf{B}^{(\mathrm{out})}}$ makes the magnetic flux density vanish at the outer boundary of the problem domain:

$$L_{\mathbf{B}^{(\mathrm{out})}}=\frac{1}{n}\sum_{i=1}^{n}inf(\mathbf{x}_i)\left(\left(B^{(\mathrm{out})}_{x_1,\mathrm{approx}}(\mathbf{x}_i)\right)^2+\left(B^{(\mathrm{out})}_{x_2,\mathrm{approx}}(\mathbf{x}_i)\right)^2\right). \quad (25)$$

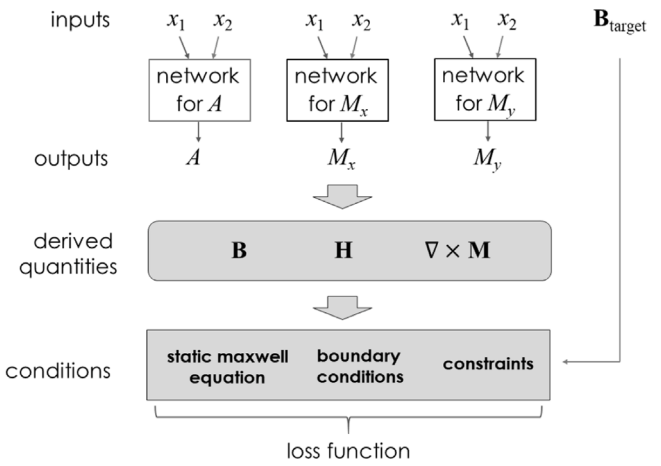

Fig. 2. Schematics of a physics informed neural network for solving magnetostatic inverse problems. Dense neural networks approximate the magnetic vector potential A and the magnetization components M_{x_1} and M_{x_2}. Inputs for the networks are the location of the points in the problem domain. From the network outputs the terms in the underlying physical equations are derived. The loss function for training the networks is the sum of squares of the residuals associated with the static Maxwell equation, boundary conditions, and constraints, which involve target values for the magnetic flux density.

Similarly to the neural network approximation of the magnetic flux density, we can derive the neural network approximation of the magnetic field inside $\mathbf{H}^{(\text{in})}_{\text{approx}}$ and outside the magnet $\mathbf{H}^{(\text{out})}_{\text{approx}}$ from the neural network approximations of the magnetic vector potential. The following two loss functions account for the continuity of the normal component of the magnetic flux density and the continuity of the tangential component of the magnetic field at the surface of the magnet:

$$L_{B_\text{n}} = \frac{1}{n}\sum_{i=1}^{n} bnd(\mathbf{x}_i)\left(\left(\mathbf{B}^{(\text{in})}_{\text{approx}}(\mathbf{x}_i) - \mathbf{B}^{(\text{out})}_{\text{approx}}(\mathbf{x}_i)\right)\cdot \mathbf{n}(\mathbf{x}_1)\right)^2, \tag{26}$$

$$L_{H_\text{t}} = \frac{1}{n}\sum_{i=1}^{n} bnd(\mathbf{x}_i)\left(\left(\mathbf{H}^{(\text{in})}_{\text{approx}}(\mathbf{x}_i) - \mathbf{H}^{(\text{out})}_{\text{approx}}(\mathbf{x}_i)\right)\times \mathbf{n}(\mathbf{x}_i)\right)^2. \tag{27}$$

During training of the neural network, the loss is minimized with the stochastic gradient descent algorithm. Input data for training are quasi-randomly sampled points from the problem domain. The training set contains N points, which are quasi-randomly sampled with a Sobol sequence [36]. For training, the points are combined into batches. In each iteration step, the stochastic gradient descent method adjusts the weights of the neural network according to the samples of one batch. The batch size $n < N$ is the number of points which are used to evaluate the loss function (19). The batch size needs to be large enough so that each batch contains points in the magnet, outside the magnet, close to the magnet's surface and close to the outer boundary [37]. During optimization, the algorithm passes several times through the complete training set. The number of passes through the entire training set is commonly referred to as epochs.

3.2. Inverse magnetostatic problems

For inverse magnetostatic problems, we introduce additional neural networks

$$M_{x_1,\text{approx}} = \mathcal{N}_{M_{x_1}}(x_1, x_2, \mathbf{w}_{M_{x_1}}), \tag{28}$$

$$M_{x_2,\text{approx}} = \mathcal{N}_{M_{x_2}}(x_1, x_2, \mathbf{w}_{M_{x_2}}), \tag{29}$$

that approximate the unknown magnetization distribution. The loss function for training is augmented with loss functions for the constraint $\left|\mathbf{M}_{\text{approx}}\right| = M_\text{s}$. The set of governing partial differential equations, boundary conditions, and constraints have to reflect the target magnetic flux density $\mathbf{B}_{\text{target}}$ of the problem. One prominent magnetostatic inverse problem is the Halbach cylinder (see Fig. 1b). We want to find the orientation of the magnetization in a long magnetic cylinder that generates a uniform vertical field in a cylindrical cavity and is zero outside the magnetic system. For simplicity we consider the cylinder to be infinitely extended and make use of translational symmetry along the cylinder axis. For the Halbach cylinder the governing equations are (8) and the interface conditions

$$\left(\mathbf{B}^{(\text{in})} - \mathbf{B}^{(\text{cavity})}_{\text{target}}\right)\cdot \mathbf{n} = 0 \text{ at the inner surface of the cylinder,} \tag{30}$$

$$\left(\mathbf{H}^{(\text{in})} - \mathbf{H}^{(\text{cavity})}_{\text{target}}\right)\times \mathbf{n} = 0 \text{ at the inner surface of the cylinder,} \tag{31}$$

$$\mathbf{B}^{(\text{in})}\cdot \mathbf{n} = 0 \text{ at the outer surface of the cylinder,} \tag{32}$$

$$\mathbf{H}^{(\text{in})}\times \mathbf{n} = 0 \text{ at the outer surface of the cylinder.} \tag{33}$$

The magnetostatic inverse problem for the Halbach cylinder leads to the following loss function

$$L_{\text{halbach}} = L_{\mathbf{M}} + L_{A^{(\text{in})}} + L_{B_\text{n},1} + L_{H_\text{t},1} + L_{B_\text{n},2} + L_{H_\text{t},2}. \tag{34}$$

We introduce the indicator functions $bnd_1(\mathbf{x})$ and $bnd_2(\mathbf{x})$ to select training points close to the inner and outer surface of the cylinder, respectively. The individual loss functions are

$$L_{\mathbf{M}} = \frac{1}{n}\sum_{i=1}^{n} in(\mathbf{x}_i)\left(\sqrt{M^2_{x_1,\text{approx}} + M^2_{x_2,\text{approx}}} - 1\right)^2, \tag{35}$$

$$L_{A^{(\text{in})}} = \frac{1}{n}\sum_{i=1}^{n} in(\mathbf{x}_i)\left(\left.\frac{\partial^2 A^{(\text{in})}_{\text{approx}}(\mathbf{x})}{\partial x_1^2}\right|_{\mathbf{x}=\mathbf{x}_i} + \left.\frac{\partial^2 A^{(\text{in})}_{\text{approx}}(\mathbf{x})}{\partial x_2^2}\right|_{\mathbf{x}=\mathbf{x}_i} - \mu_0\left(\left.\frac{\partial M_{x_1,\text{approx}}(\mathbf{x})}{\partial x_2}\right|_{\mathbf{x}=\mathbf{x}_i} - \left.\frac{\partial M_{x_2,\text{approx}}(\mathbf{x})}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_i}\right)\right)^2, \tag{36}$$

$$L_{B_\text{n},1} = \frac{1}{n}\sum_{i=1}^{n} bnd_1(\mathbf{x}_i)\left(\left(\mathbf{B}^{(\text{in})}_{\text{approx}}(\mathbf{x}_i) - \mathbf{B}^{(\text{cavity})}_{\text{target}}(\mathbf{x}_i)\right)\cdot \mathbf{n}(\mathbf{x}_i)\right)^2, \tag{37}$$

$$L_{H_\text{t},1} = \frac{1}{n}\sum_{i=1}^{n} bnd_1(\mathbf{x}_i)\left(\left(\mathbf{H}^{(\text{in})}_{\text{approx}}(\mathbf{x}_i) - \mathbf{H}^{(\text{cavity})}_{\text{target}}(\mathbf{x}_i)\right)\times \mathbf{n}(\mathbf{x}_i)\right)^2, \tag{38}$$

$$L_{B_\text{n},2} = \frac{1}{n}\sum_{i=1}^{n} bnd_2(\mathbf{x}_i)\left(\mathbf{B}^{(\text{in})}_{\text{approx}}(\mathbf{x}_i)\cdot \mathbf{n}(\mathbf{x}_1)\right)^2, \tag{39}$$

$$L_{H_\text{t},2} = \frac{1}{n}\sum_{i=1}^{n} bnd_2(\mathbf{x}_i)\left(\mathbf{H}^{(\text{in})}_{\text{approx}}(\mathbf{x}_i)\times \mathbf{n}(\mathbf{x}_i)\right)^2. \tag{40}$$

The schematics of a physics informed neural network for the solution of magnetostatic inverse problems is given in Fig. 2.

In inverse modeling regularization terms ensure the smoothness of the solution [5]. However, we found that such an explicit regularization is not required for solving inverse magnetostatic problems with physics informed neural networks. We use neural networks with the hyperbolic tangent as activation function for the hidden layers. The neural network approximations of the magnetization components (28) and (29) are smooth functions. In addition, our networks are simple containing only a few hidden layers. We speculate that the architecture takes the role of the regularization term commonly used in inverse modeling.

3.3. Ritz based micromagnetics

In this section we show how to solve micromagnetic problems with the deep Ritz method. In order to introduce the methodology, we first show that the deep Ritz method as introduced by E and Yu [17] can be applied as an alternative to the collocation method described in Section 3.1, for the solution of the magnetostatic field problem. Similar to the Ritz method in the context of the finite element method [22] we minimize the magnetostatic energy functional (15) by using it as loss function for training a neural network. With the deep Ritz method for solving magnetostatic problems we use a single neural network

$$A_{\text{approx}} = \mathcal{N}_A(x_1, x_2, \mathbf{w}_A) \tag{41}$$

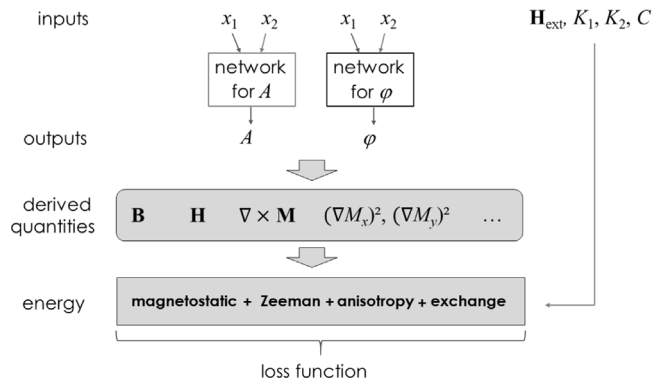

Fig. 3. Schematics of a physics informed neural network for solving micromagnetic problems. Dense neural networks approximate the magnetic vector potential A and the magnetization angle φ. Inputs for the networks are points in the problem domain. From the network outputs the terms in the micromagnetic energy densities are derived. The loss function for training the networks is the total Gibbs free energy, which is evaluated by quasi-Monte-Carlo integration.

for the approximation of the magnetic vector potential. The weights and biases, which are represented by the vector $\mathbf{w}_A$, are determined during training of the network by minimizing the functional (15). In order to evaluate this integral, we apply Monte-Carlo integration with quasi-randomly sampled points $\mathbf{x}_i$. The use of quasi-random points for Monte-Carlo integration improves convergence [38] since clumps of points that occur for random sampling can be avoided. Similarly, Hennig and co-workers [18] apply quasi-Monte Carlo integration to evaluate the integrals occurring during the solution of partial differential equations with physics informed neural networks. The loss function for training of the neural network is

$$L_{\text{mag}} = \frac{V_{\text{dom}}}{n}\frac{1}{2}\sum_{i=1}^{n}\left(\left(\left.\frac{\partial A_{\text{approx}}(\mathbf{x})}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_i}\right)^2 + \left(\left.\frac{\partial A_{\text{approx}}(\mathbf{x})}{\partial x_2}\right|_{\mathbf{x}=\mathbf{x}_i}\right)^2 + 2\mu_0\, in(\mathbf{x}_i)\left(M_{x_2}(\mathbf{x}_i)\left.\frac{\partial A_{\text{approx}}(\mathbf{x})}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_i} - M_{x_1}(\mathbf{x}_i)\left.\frac{\partial A_{\text{approx}}(\mathbf{x})}{\partial x_2}\right|_{\mathbf{x}=\mathbf{x}_i}\right)\right) \tag{42}$$

Here V_{dom} is the area of the problem domain.

In micromagnetics we want to compute the magnetization $\mathbf{M}$. In addition to the neural network (41) we introduce an additional neural network

$$\varphi_{\text{approx}} = \mathcal{N}_{\varphi}(x_1, x_2, \mathbf{w}_{\alpha}) \tag{43}$$

for the angle of the magnetization with respect to the x_1 axis. In two-dimensional micromagnetics the magnetization is restricted to lie in the (x_1, x_2) plane. Therefore, the neural network approximation of the magnetization components are given by

$$M_{x_1,\text{approx}} = M_{\text{s}}\cos(\varphi_{\text{approx}}), \tag{44}$$

$$M_{x_2,\text{approx}} = M_{\text{s}}\sin(\varphi_{\text{approx}}). \tag{45}$$

The total loss function is the sum

$$L = L_{\text{mag}} + L_{\text{zee}} + L_{\text{ani}} + L_{\text{ex}}. \tag{46}$$

The summands represent the magnetostatic energy, the Zeeman energy, the anisotropy energy, and the exchange energy. These energies are again evaluated with quasi-Monte-Carlo integration. The schematics of a physics informed neural network for micromagnetic simulations is shown in Fig. 3. In addition to L_{mag}, the loss functions are

$$L_{\text{zee}} = \frac{\mu_0 V_{\text{dom}}}{n}\sum_{i=1}^{n} in(\mathbf{x}_i)\left(-\mu_0 M_{x_1,\text{approx}}(\mathbf{x}_i)H_{\text{ext},x_1} - \mu_0 M_{x_2,\text{approx}}(\mathbf{x}_i)H_{\text{ext},x_2}\right), \tag{47}$$

$$L_{\text{ani}} = \frac{\mu_0 V_{\text{dom}}}{n}\sum_{i=1}^{n} in(\mathbf{x}_i)\left(K_1\left(\frac{M_{x_1,\text{approx}}(\mathbf{x}_i)}{M_{\text{s}}}\right)^2 + K_2\left(\frac{M_{x_1,\text{approx}}(\mathbf{x}_i)}{M_{\text{s}}}\right)^4\right), \tag{48}$$

$$L_{\text{ex}} = \frac{\mu_0 V_{\text{dom}}}{n}\sum_{i=1}^{n} in(\mathbf{x}_i)\left(\frac{C}{M_{\text{s}}^2}\left[\left(\left.\nabla M_{x_1,\text{approx}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}_i}\right)^2 + \left(\left.\nabla M_{x_2,\text{approx}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}_i}\right)^2\right]\right). \tag{49}$$

Please note that the energies are scaled with the factor μ_0. Here, we assumed that the anisotropy axis is parallel to the x_2 direction.

4. Results

We apply the Keras/Tensorflow wrapper SciAnn [37] for implementing physics informed neural networks. We sample training points quasi-uniformly in the problem domain, applying the Sobol sequence as implemented in the Python library scikit-optimize [39]. This procedure for finding the Cartesian coordinates of the training points is applied for all the numerical experiments reported in this work. The approximation by a physics informed neural network converges to the solution of the partial differential equation with increasing number of training points N [40,41]. We treated N as a hyperparameter of the model. We increased N and evaluated the energy of the system by Monte Carlo integration with test points sampled independently of the training points. For sufficiently large N, no noticeable decrease in energy was observed.

4.1. Magnetic field of a uniformly magnetized infinite prism

As test case, we pick a classical problem in micromagnetics [24]. We compute the magnetic field for a uniformly magnetized particle with rectangular cross section using the collocation based physics informed neural networks and the deep Ritz method. We consider a magnetic prism infinitely extended in the x_3 direction. It is uniformly magnetized in x_2 direction with the magnetization M_{s}. The lower left corner and upper right corner of the rectangle are $(x_{1,\min}, x_{2,\min})$ and $(x_{1,\max}, x_{2,\max})$, respectively. The magnetic induction $\mathbf{B} = \mu_0\mathbf{H} + \mu_0\mathbf{M}$ inside the magnet is [24]

$$B_{x_1} = -\frac{\mu_0 M_{\text{s}}}{4\pi}\ln \times \frac{\left((x_1 - x_{1,\min})^2 + (x_2 - x_{2,\min})^2\right)\left((x_1 - x_{1,\max})^2 + (x_2 - x_{2,\max})^2\right)}{\left((x_1 - x_{1,\min})^2 + (x_2 - x_{2,\max})^2\right)\left((x_1 - x_{1,\max})^2 + (x_2 - x_{2,\min})^2\right)}, \tag{50}$$

$$B_{x_2} = -\frac{\mu_0 M_{\text{s}}}{4\pi}\left(\arctan\frac{x_1 - x_{1,\min}}{x_2 - x_{2,\min}} - \arctan\frac{x_1 - x_{1,\min}}{x_2 - x_{2,\max}} - \arctan\frac{x_1 - x_{1,\max}}{x_2 - x_{2,\min}} + \arctan\frac{x_1 - x_{1,\max}}{x_2 - x_{2,\max}}\right) + \mu_0 M_{\text{s}}. \tag{51}$$

We set the corners $(x_{1,\min}, x_{2,\min})$ and $(x_{1,\max}, x_{2,\max})$ to $(-0.5, -0.5)$ and $(0.5, 0.5)$, respectively.

The magnetostatic energy density of the infinite prism [22] is $\mu_0 M_{\text{s}}^2/4$. This result can be derived from the theorem that the magnetostatic self-energy of a uniformly magnetized particle of arbitrary shape is equivalent to that of suitably chosen ellipsoid of the same volume [42] and the demagnetizing factor $D = 1/2$ of an infinite prism with square cross-section [43]. For our simulations, we set $\mu_0 M_{\text{s}} = 1$ T.

The problem domain extends outside the magnetic region. It is a square ranging from (−5,−5) to (5,5). In other words, we truncate the problem domain for the magnetic vector potential at $x_1 = \pm 5$ and $x_2 = \pm 5$. All networks have identical layout with 8 hidden layers with 32 neurons each. We used tanh as activation function.

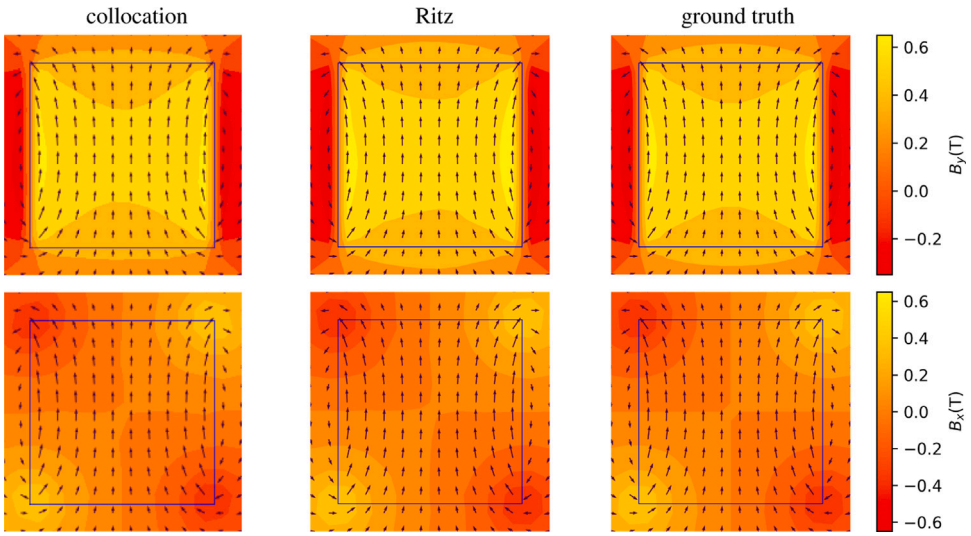

Fig. 4. Demagnetizing field for the uniform magnetic particle with square cross-section. Left: Collocation method. Center: Ritz method. Right: Analytical solution.

Table 1
Comparison of physics informed neural networks based on the collocation method and the Ritz method for solving a magnetostatic problem. In last line give the number of passes through the entire training set (epochs). At the given number of epochs the learning rate reached its minimum value.

Quantity	Equation	Collocation method	Ritz method
Magnetostatic energy	$-(\mu_0/2)\int_{V^{(\mathrm{in})}} \mathbf{H}\cdot\mathbf{M}\mathrm{d}^2x$	$0.255\mu_0 M_s^2 V^{(\mathrm{in})}$	$0.248\mu_0 M_s^2 V^{(\mathrm{in})}$
Mean absolute error	$(1/m)\sum_i^m \left\|\mathbf{B}_{\mathrm{true}}(\mathbf{x}_i) - \mathbf{B}_{\mathrm{approx}}(\mathbf{x}_i)\right\|$	0.014 T	0.011 T
Epochs	When learning rate $< 10^{-8}$	604	527

The total number of sampling points is $N = 2^{24}$. We split the training set into 2^6 batches of size $n = 2^{18}$. For the indicator *in* function we used

$$in(x_1, x_2) = \left(\frac{\mathrm{sgn}(x_1 + 0.5)}{2} - \frac{\mathrm{sgn}(x_1 - 0.5)}{2}\right) \times \left(\frac{\mathrm{sgn}(x_2 + 0.5)}{2} - \frac{\mathrm{sgn}(x_2 - 0.5)}{2}\right). \quad (52)$$

The indicator functions *bnd* and *inf* were set to 1 when the distance of a point to the respective boundary was less than 10^{-3}.

For optimization, we applied the Adam method [44] with an initial step size of 10^{-3}. The number of complete passes through the training set (epochs) was 2000. However, this value is not reached owing to the following early stopping method. When there is no decrease in the loss function for ten epochs the learning rate is reduced by 1/2. Reducing the step size (learning rate) in a stochastic gradient descent method reduces the fluctuations in the loss. The different batches will give different gradients of the loss with respect to the weights which in turn causes random oscillations in the loss. A gentle decrease of the learning rate helps convergence [45]. Training is stopped if the learning rate reaches $\eta_{\min} = 10^{-8}$. The number of epochs at which $\eta_{\min}$ is reached is given in Table 1.

Fig. 4 shows the computed magnetic flux density of a uniformly magnetized particle with rectangular cross-section. Virtually there is no difference between the results obtained from the collocation based physics informed neural network and deep Ritz method. A detailed comparison is given in Table 1. The magnetostatic energy is computed by quasi-Monte-Carlo integration over the volume of the magnetic region $V^{(\mathrm{in})}$. For integration 10^5 points sampled with Sobol distribution are used. The same points are used to compute the mean absolute error.

4.2. Halbach cylinder

We test the use of physics informed neural networks for the solution of a classical inverse magnetostatic problem. This can be achieved with permanent magnet flux sources with a well defined arrangement of permanent magnets [1]. We want to use the methodology outlined in Section 3.2, to compute the orientation of the magnetization in a long magnetic cylinder that generates a uniform vertical field in a cylindrical cavity and is zero outside the magnetic system.

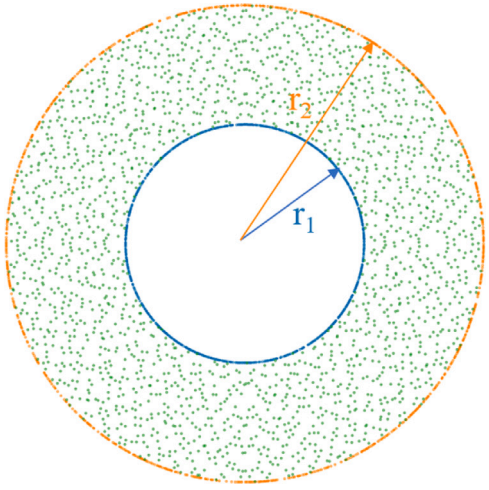

Fig. 5. Points sampled for training of the physics informed neural network. The three set of points are sampled in the ring and at the inner and outer surface of the cylinder. For visibility the number of points was reduced as compared to the actual data used for training the network.

The analytic solution is well-known [25]. Let θ denote the angle with the vertical axis. The inner radius and the outer radius of the magnetic hollow cylinder are r_1 and r_2. A uniform field is achieved when the magnetization $\mathbf{M}$ at any position θ in the magnet is rotated by the angle 2θ with respect to the vertical axis. The uniform flux density in the cavity is $B = \mu_0 M_s \ln(r_2/r_1)$. We set $\mu_0 M_s = 1$ T.

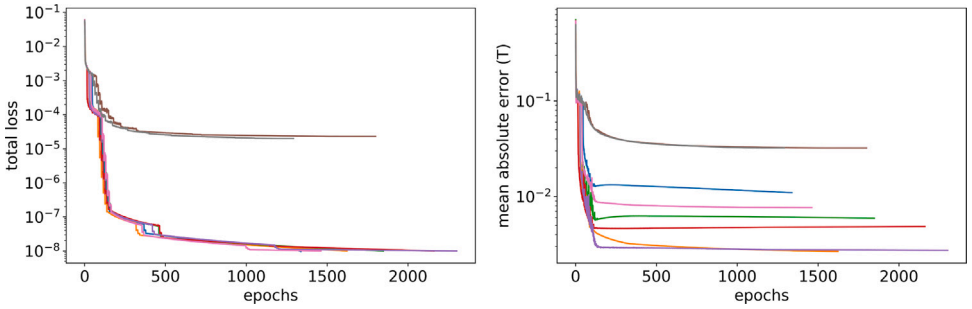

Fig. 6. Decrease of the total loss (left hand side) and the mean absolute error (right hand side) during training. L_{halbach} and $(\mu_0/m)\sum_i^m \left|\mathbf{M}_{\text{true}}(\mathbf{x}_i) - \mathbf{M}_{\text{approx}}(\mathbf{x}_i)\right|$ are plotted as a function of the number of epochs for eight different runs with the parameters $h \times l = 8 \times 32$ and $n = 2^8$.

Table 2
Minimal loss L_{halbach} achieved with different hyperparameters. h is the number of hidden layers, l is the number of neurons per hidden layer, and n is the batch size. For each set of hyperparameters L_{halbach} and the number of passes through the training set (epochs) is given. The total number of training points was $N = 2^{20}$. The last column gives the mean absolute error, $(\mu_0/m)\sum_i^m \left|\mathbf{M}_{\text{true}}(\mathbf{x}_i) - \mathbf{M}_{\text{approx}}(\mathbf{x}_i)\right|$, computed with about 2.3×10^5 quasi-randomly sampled points. Owing to the inherent randomness of the algorithm slightly different values are obtained when the simulation is repeated for the same set of hyperparameters. The third column is a consecutive number for several runs with the same parameters.

Layout $h \times l$	Batch size n	Run	Epochs	Loss L_{halbach}	Mean absolute error (T)
2×32	2^{10}	1	2421	3.5×10^{-4}	0.089
4×16	2^{14}	1	10 225	7.5×10^{-4}	0.13
4×32	2^8	1	2233	2.8×10^{-5}	0.021
4×32	2^{10}	1	1475	9.8×10^{-5}	0.055
4×32	2^{14}	1	8320	4.7×10^{-4}	0.1
8×32	2^8	1	1462	1.0×10^{-8}	0.0076
8×32	2^8	2	1292	2.0×10^{-5}	0.032
8×32	2^8	3	1848	1.0×10^{-8}	0.0059
8×32	2^8	4	1338	9.6×10^{-9}	0.011
8×32	2^8	5	2302	1.0×10^{-8}	0.0027
8×32	2^8	6	1623	1.0×10^{-8}	0.0027
8×32	2^8	7	2161	1.0×10^{-8}	0.0048
8×32	2^8	8	1800	2.3×10^{-5}	0.032
8×32	2^{10}	1	1840	3.3×10^{-5}	0.028
8×32	2^{14}	1	5752	3.0×10^{-4}	0.1

The inputs for the three networks $\mathcal{N}_A$, $\mathcal{N}_{M_{x_1}}$, and $\mathcal{N}_{M_{x_2}}$ were the position of the training points. The total number of points in the training set, N, is $N = 2^{20}$. It consists of three distinct set of points. The points in the ring, points at the inner surface, and points at the outer surface. The indicator functions in, bnd_1, and bnd_2 were set accordingly. $N/2$ points are sampled quasi-randomly in an annulus with inner radius $r_1 = 1$ and outer radius $r_2 = 2$. We first create a Sobol sequence of quasi-random points in a square and then map the points [46] to the annulus. In addition, we create $N/4$ randomly sampled points on the outer surface and $N/4$ randomly sampled points at the inner surface. Fig. 5 shows the distribution of the training points. For visibility the number of points is lower than that actually used for training.

We tuned the hyperparameters of the network with a manual search. Table 2 compares the minimal value of the loss (34) achieved with different sets of hyperparameters. We modified the layout of the network $h \times l$ and the batch size n. Here h is the number of hidden layers that contain l neuron each. For optimization, we applied the Adam method [44] with an initial step size of 10^{-3}. We applied early stopping by the learning rate as described in Section 4.1 with a minimum learning rate of $\eta_{\min} = 10^{-10}$. The number of maximum possible epochs was set to a very high number. The training was stopped when the minimum learning rate was reached or the loss dropped below 10^{-8}. Table 2 also lists the mean absolute error in the magnetization. The mean absolute error $(\mu_0/m)\sum_i^m \left|\mathbf{M}_{\text{true}}(\mathbf{x}_i) - \mathbf{M}_{\text{approx}}(\mathbf{x}_i)\right|$ was computed with about 2.3×10^5 quasi-randomly sampled points. Please note that we did not use the mean absolute error for tuning of the hyperparameters because for general problems the solution is not available. Instead, we relied on the total loss to select the network topology and the batch size.

The results listed in Table 2 shows that reducing the batch size for fixed network layout reduces the total loss. This may be explained by the observations of Keskar and co-workers [47] who show that training with a large batch size finds minima which are much closer to the initial state than training with a smaller batch size. Methods with a smaller batch size explore the energy landscape and move from the initial point towards minima that are located farther away. The minimum loss was obtained with $h \times l = 8 \times 32$ and a batch size of $n = 2^8$.

Repeated simulations with the same set of hyperparameters show different results [48,49] owing to the inherent randomness of the stochastic gradient descent method. There are two reasons for randomness: The random initialization of the weights and randomly shuffled training sets. At the start of the algorithm the initial weights are randomly set by a truncated normal distribution. Each batch contains randomly picked points from the total training set. The gradients of the loss with respect to the weights will fluctuate from batch to batch when passing through the training data. Fig. 6 shows the total loss and the mean absolute error in the magnetization as a function of the number of full passes through the training set (epochs) for different runs with $h \times l = 8 \times 32$ and $n = 2^8$. We observe a variance between the different runs. Loss and error drop rapidly at the beginning of the training for six out of eight runs. For two runs, the systems seems to be trapped in a bad local minimum. The use of adaptive activation functions [50,51] can mitigate this problem. In practice, we can train the neural network several times with different initial seeds and use the weights that result in the lowest total loss for production runs.

Fig. 7 compares the analytic solution and the estimate of the neural network for the Halbach cylinder computed with the hyperparameters $h \times l = 8 \times 32$, $n = 2^8$ for run = 1 at epoch = 1462. The vector plots show the magnetization in the hollow cylinder. The mean absolute error is 0.008 T. Fig. 8 shows the distribution of the error within the ring. Almost everywhere the error is below 0.01 T. A higher error occurs near the outer surface where the magnetization is nearly vertical. We speculate that a different split of the training points between inside, inner, and outer surface might improve the error distribution.

4.3. Magnetization reversal of an infinite prism

Here we apply the deep Ritz method for computing magnetization reversal of a magnetic particle as outlined in Section 3.3. The magnetization angle and the magnetic vector potential are approximated with two dense neural networks $\mathcal{N}_\varphi(x_1, x_2, \mathbf{w}_\varphi)$ and $\mathcal{N}_A(x_1, x_2, \mathbf{w}_A)$. The algorithmic framework for training neural networks is used to minimize the total Gibbs free energy for points along the demagnetization curve.

Starting from a strong external field that saturates the particle, the repeated minimization of the Gibbs free energy gives the magnetic states along the demagnetization curve [22]. The pretrained weights $\mathbf{w}_\varphi$ and $\mathbf{w}_A$ from the previous field step are used as initial weights for

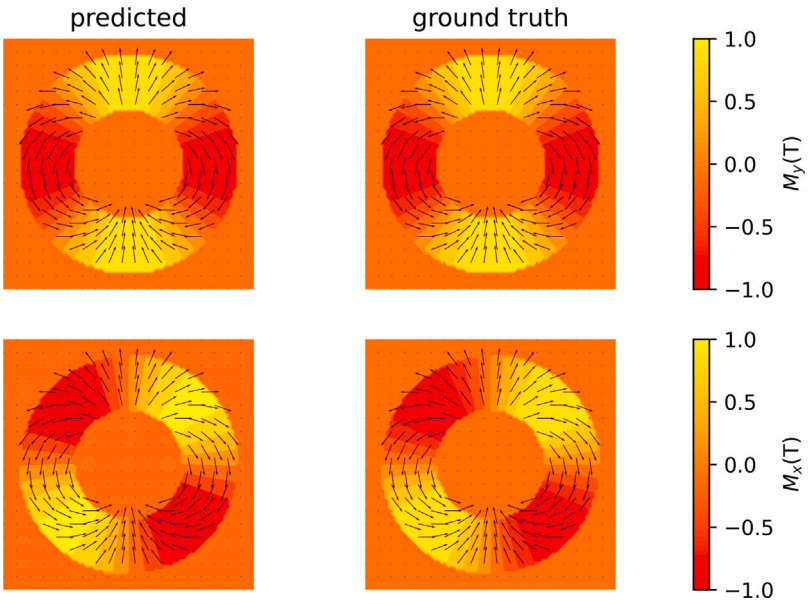

Fig. 7. Comparison of the estimate from physics informed neural network and the analytical solution for a classical inverse magnetostatic problem. The computed magnetization for a Halbach cylinder is shown for $h \times l = 8 \times 32$ and $n = 2^8$ for run = 1 at epoch = 1462.

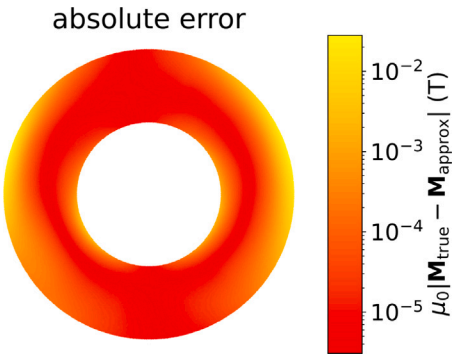

Fig. 8. Absolute error $\mu_0 \left|\mathbf{M}_{\text{true}}(\mathbf{x}) - \mathbf{M}_{\text{approx}}(\mathbf{x})\right|$ between the magnetization of the true solution and the neural network approximation for the Halbach cylinder for $h \times l = 8 \times 32$ and $n = 2^8$ for run = 1 at epoch = 1462.

the successive training at the slightly decreased external field. In order to initialize the weights we used two steps. Firstly, we apply a strong external field and minimize the Zeeman energy by adjusting only $\mathbf{w}_\varphi$. Secondly, we keep $\mathbf{w}_\varphi$ fixed and minimize the magnetostatic energy by adjusting $\mathbf{w}_A$. For these two training steps, we apply the early stopping method discussed above. After this initial training, the networks $\mathcal{N}_\varphi$ and $\mathcal{N}_A$ give the magnetization angle and magnetic vector potential of the saturated state, respectively. In what follows we minimize the upper bound (16) for total energy for decreasing values of the external field by adjusting $\mathbf{w}_\varphi$ and $\mathbf{w}_A$ simultaneously.

The demagnetization curve of a small hard magnetic particle has three characteristic branches. Initially the magnetization starts to deviate from the easy axis by reversible rotations. Especially near corners the magnetization rotates to minimize the magnetostatic energy [24]. The typical flower state is formed [52]. During reversible processes the system follows a path of subsequent local minima [33]. At a critical value of the external field irreversible switching occurs. The system escapes from a saddle point towards the next minimum of the energy [53]. Irreversible switching leads to the lower branch of the hysteresis loop.

Optimizers used for training neural networks are designed to search for a deep local or global minimum of the loss function. In contrast, for computing hysteresis we want to follow a local minimum closely without escaping over a non-zero energy barrier. When computing the successive magnetic states along the demagnetization curve the energy should never increase for a fixed external field. On the other hand, once a saddle point is reached, we want to get out of the saddle point immediately.

To meet the first requirement we modify the standard training method and apply an early stopping algorithm [54]. We discard the current state, stop training, and move to the next field whenever an increase of the energy occurs during training.

To get out of saddle points we use root mean square propagation (RMSprop) [45] as optimizer. This is an adaptive gradient method which adapt the search direction by scaling the gradient. Let $g_i = \partial L/\partial w_i$ the component of the gradient of the loss function with respect to the weight w_i. The intuition behind RMSprop is to use an approximation of the sign of g_i instead of g_i as search direction. Then it is easy to move out regions with tiny gradients [45]. Let us denote the moving average of the squared gradient of weight i with $MeanSquare_{i,t}$. Then the update rule of the t's iteration is as follows:

$$MeanSquare_{i,t} = \rho\, MeanSquare_{i,t-1} + (1-\rho) g_i^2, \tag{53}$$

$$w_{i,t+1} = w_{i,t} + \eta \frac{g_i}{\sqrt{MeanSquare_{i,t}} + \epsilon}. \tag{54}$$

Here η is the learning rate, ρ the discounting factor, and t the iteration count. The total number of training points is split into batches. If N is the number of training bounds and n is the batch size, there are N/n iterations to complete a full pass through the training set (epoch). The moving average of squared gradients in (53) smooths the mean square over adjacent batches. The parameter ϵ is regarded as a regularization term to avoid a large step when $\sqrt{MeanSquare_{i,t}}$ is close to zero. However, it also controls the adaptivity level [48]. Large values of ϵ reduce the influence of $MeanSquare_{i,t}$ and makes the algorithm more like stochastic gradient descent [48,55]. Please note that $g_i/\left(\sqrt{MeanSquare_{i,t}} + \epsilon\right)$ is an approximation of the sign of the ith component of the gradient, $\text{sgn}(g_i) = g_i/\sqrt{g_i^2}$.

Treating RMSprop as a preconditioned stochastic gradient descent method, Staib and co-workers [55] found an optimal relation between the learning rate η and the discounting factor ρ:

$$\rho_k = 1 - c\eta_k^{2/3}. \tag{55}$$

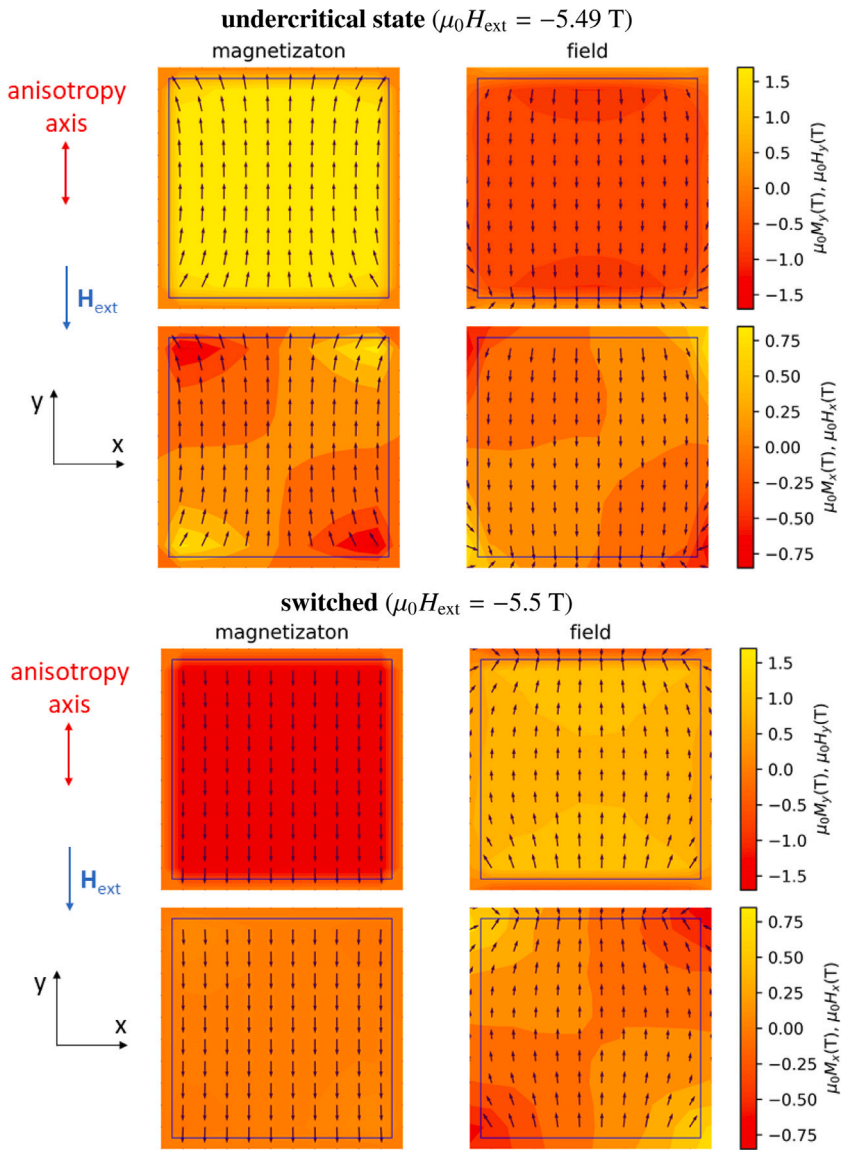

Fig. 9. Neural network approximation of the magnetization and the demagnetizing field before and after irreversible switching of a $Nd_2Fe_{14}B$ particle with square cross-section. The particle is infinitely extended in the direction perpendicular to the drawing plane. The magneto-crystalline anisotropy axis is parallel to the y-axis. The external field is antiparallel to the y-axis as indicated by the minus sign. The sub-panel labeled "undercritical state" gives the system before switching. The sub-panel labeled "swichted" gives the system after irreversible switching. The arrows in the left column give the magnetization. The arrows in the right column give the demagnetizing field. The color code of the top row of each sub-panel gives the y-component of the magnetization (left) and the demagnetizing field (right). The color code of the bottom row of each sub-panel gives the x-component of the magnetization (left) and the demagnetizing field (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

They also suggest using a decreasing step size. Therefore, we use $\eta_k = \eta_0/\sqrt{k}$. Here k counts the full passes through the training set (epochs). We set the initial learning rate to $\eta_0 = 10^{-3}$. We tuned the parameters c and ϵ by comparing the computed switching field with the analytic result for a hard magnetic cylinder with circular cross-section.

We simulate the reversal of $Nd_2Fe_{14}B$ particles infinitely extended in a direction perpendicular to the particle's anisotropy axis [22]. The intrinsic material parameters used for the simulations were $\mu_0 M_s$ = 1.61 T, $K_1 = 4.3$ MJ/m^3, $K_2 = 0.65$ MJ/m^3, and $C = 7.7$ pJ/m [56]. The field step for the simulations was $\Delta\mu_0 H_{ext} = 0.01$ T. For a small particle with circular cross-section the demagnetizing field is uniform. For an external field applied parallel to the magneto-crystalline anisotropy axis the irreversible switching field is $2K_1/(\mu_0 M_s)$ [57]. This analytical value is $\mu_0 H_{sw,true} = 6.71$ T for $Nd_2Fe_{14}B$. We computed the switching field using deep neural networks as outlined in Section 3.3. The total number of distinct training points was $N = 2^{24}$ and the batch size was $n = 2^{18}$. The dense neural networks had 8 hidden layers with 32 neurons each.

We used simulations of the switching field for a cylinder with a diameter of 4.5 nm for tuning the hyperparameters c and ϵ. We found that too small a value of c caused premature switching: For $c < 1$ the magnet reverses at an external field $|H_{ext}| < H_{sw,true}$ during the first pass through the training set, which indicates an unwanted escape from a local minimum. For $c = 10$ and $\epsilon = 10^{-7}$ the computed switching field, $\mu_0 H_{sw,approx} = 6.74$ T. It is slightly higher than the analytical value, but the relative error is only 0.4 percent.

We computed magnetization reversal for a $Nd_2Fe_{14}B$ particle with a square cross-section. The particle is infinitely extended in z-direction which points out of the drawing plane in Fig. 9. The anisotropy axis is parallel to the y-axis. The external field is applied parallel to the anisotropy direction. The demagnetization curve (not shown) is almost perfectly square with a single step at the switching field. Before

switching the magnetization tilts near the corners in order to reduce the magnetostatic energy. By this process the well-known flower state is formed [22,24,52]. After switching the magnetization is antiparallel to the y-axis. For a particle size of 60×60 nm^2 in xy-direction, the irreversible switching field computed with finite element micromagnetic simulations is $\mu_0 H_{\mathrm{sw,fem}} = 5.47$ T [22]. The optimization of the neural network is a stochastic algorithm. Repeated simulations show slight fluctuations in the switching field approximated with the deep neural network. For $c = 10$ and $\epsilon = 10^{-7}$, the switching fields from repeated runs varied in the range from $\mu_0 H_{\mathrm{sw,approx}} = 5.5$ T to $\mu_0 H_{\mathrm{sw,approx}} = 5.52$ T. The maximum relative error with respect to the finite element result was 0.9 percent. Fig. 9 shows the computed magnetization and the demagnetizing field just before and after irreversible switching. The flower state is clearly seen in the undercritical state before switching.

5. Conclusion

We demonstrated the use of physics informed neural networks (PINNs) for magnetostatics, micromagnetics, and hysteresis computation. We used a deep neural network to approximate the magnetic vector potential. Training the neural network reduces the residuals of the static Maxwell equation at randomly sampled points in the problem domain, at its boundary and at material interfaces. To account for the interface condition of the magnetic flux density and the magnetic field at the surface of a magnetic material, separate networks for different regions were introduced [34]. For the solution of inverse magnetostatic problems, we introduced additional neural networks that estimate the unknown magnetization. The loss function contains additional terms that penalize deviations from target conditions. The methodology was tested for the computation of the magnetization distribution in Halbach cylinders [1].

Using Brown's upper bound for the magnetostatic energy, a deep Ritz method [17] can be applied to solve magnetostatic field problems. The magnetostatic energy is the loss function for training the neural network. Adding the Zeeman energy, the ferromagnetic exchange energy density, and the magneto-crystalline anisotropy energy density, we built a micromagnetic solver that uses the algorithmic frame work of neural networks. Classical numerical schemes for micromagnetics are based on the very same energy functional [20–22].

We believe that physics informed neural networks have great potential in computational magnetics. In particular, physics informed neural networks will come with some advantages: (1) There is no need for mesh generation. (2) Inverse problems may be solved effectively [18]. (3) A whole family of problems may be solved with a single neural network [29].

CRediT authorship contribution statement

Alexander Kovacs: Software. **Lukas Exl:** Methodology, Validation, Writing – review & editing. **Alexander Kornell:** Software. **Johann Fischbacher:** Data curation. **Markus Hovorka:** Investigation. **Markus Gusenbauer:** Investigation. **Leoni Breth:** Writing – review & editing. **Harald Oezelt:** Visualization. **Dirk Praetorius:** Writing – review & editing. **Dieter Suess:** Writing – review & editing. **Thomas Schrefl:** Writing – original draft, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, Austria and the Christian Doppler Research Association, Austria is gratefully acknowledged. L.E. and D.P. acknowledge support by the Austrian Science Fund (FWF) under grant No. P31140-N32 and grant No. F65, respectively.

References

[1] K. Halbach, Application of permanent magnets in accelerators and electron storage rings, J. Appl. Phys. 57 (8) (1985) 3605–3608.
[2] M. Bashir, T. Schrefl, J. Dean, A. Goncharov, G. Hrkac, D. Allwood, D. Suess, Head and bit patterned media optimization at areal densities of 2.5 Tbit/in2 and beyond, J. Magn. Magn. Mater. 324 (3) (2012) 269–275.
[3] C. Huber, C. Abert, F. Bruckner, M. Groenefeld, S. Schuschnigg, I. Teliban, C. Vogler, G. Wautischer, R. Windl, D. Suess, 3D printing of polymer-bonded rare-earth magnets with a variable magnetic compound fraction for a predefined stray field, Sci. Rep. 7 (1) (2017) 1–8.
[4] N. Takahashi, K. Akiyama, D. Miyagi, Y. Kanai, Advanced optimization of standard head model with higher writing field and higher field gradient using 3-D ON/OFF method, IEEE Trans. Magn. 44 (6) (2008) 966–969.
[5] C. Abert, C. Huber, F. Bruckner, C. Vogler, G. Wautischer, D. Suess, A fast finite-difference algorithm for topology optimization of permanent magnets, J. Appl. Phys. 122 (11) (2017) 113904.
[6] A. Kovacs, H. Oezelt, S. Bance, J. Fischbacher, M. Gusenbauer, F. Reichel, L. Exl, T. Schrefl, M. Schabes, Numerical optimization of writer geometries for bit patterned magnetic recording, J. Appl. Phys. 115 (17) (2014) 17B704.
[7] F. Bruckner, C. Abert, G. Wautischer, C. Huber, C. Vogler, M. Hinze, D. Suess, Solving large-scale inverse magnetostatic problems using the adjoint method, Sci. Rep. 7 (1) (2017) 1–7.
[8] W.F. Brown, Micromagnetics, Vol. 18, interscience publishers, 1963.
[9] J. Fischbacher, A. Kovacs, M. Gusenbauer, H. Oezelt, L. Exl, S. Bance, T. Schrefl, Micromagnetics of rare-earth efficient permanent magnets, J. Phys. D: Appl. Phys. 51 (19) (2018) 193002.
[10] L. Exl, D. Suess, T. Schrefl, Micromagnetism, in: M. Coey, S. Parkin (Eds.), Handbook of Magnetism and Magnetic Materials, Springer International Publishing, Cham, 2020, pp. 1–44, http://dx.doi.org/10.1007/978-3-030-63101-7_7-1.
[11] A. Khan, V. Ghorbanian, D. Lowther, Deep learning for magnetic field estimation, IEEE Trans. Magn. 55 (6) (2019) 1–4.
[12] A. Kovacs, J. Fischbacher, H. Oezelt, M. Gusenbauer, L. Exl, F. Bruckner, D. Suess, T. Schrefl, Learning magnetization dynamics, J. Magn. Magn. Mater. 491 (2019) 165548.
[13] L. Exl, N.J. Mauser, T. Schrefl, D. Suess, Learning time-stepping by nonlinear dimensionality reduction to predict magnetization dynamics, Commun. Nonlinear Sci. Numer. Simul. 84 (2020) 105205.
[14] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[15] A. Koryagin, R. Khudorozkov, S. Tsimfer, Pydens: A python framework for solving differential equations with neural networks, 2019, arXiv preprint arXiv:1909.11544.
[16] E. Kharazmi, Z. Zhang, G.E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, 2019, arXiv preprint arXiv:1912.00873.
[17] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Statist. 6 (1) (2018) 1–12.
[18] O. Hennigh, S. Narasimhan, M.A. Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, S. Choudhry, Nvidia SimNet^{TM}: An AI-accelerated multi-physics simulation framework, in: International Conference on Computational Science, Springer, 2021, pp. 447–461.
[19] E. Guancial, S. DAsGUPTA, Three-dimensional finite element program for magnetic field problems, IEEE Trans. Magn. 13 (3) (1977) 1012–1015.
[20] P. Asselin, A. Thiele, On the field Lagrangians in micromagnetics, IEEE Trans. Magn. 22 (6) (1986) 1876–1880.
[21] D. Fredkin, T. Koehler, Numerical micromagnetics by the finite element method, IEEE Trans. Magn. 23 (5) (1987) 3385–3387.
[22] T. Schrefl, J. Fidler, H. Kronmüller, Nucleation fields of hard magnetic particles in 2D and 3D micromagnetic calculations, J. Magn. Magn. Mater. 138 (1–2) (1994) 15–30.
[23] C.W. Steele, Numerical Computation of Electric and Magnetic Fields, Springer Science & Business Media, 2012.
[24] M. Grönefeld, H. Kronmüller, Calculation of strayfields near grain edges in permanent magnet material, J. Magn. Magn. Mater. 80 (2–3) (1989) 223–228.
[25] K. Halbach, Design of permanent multipole magnets with oriented rare earth cobalt material, Nucl. Instrum. Methods 169 (1) (1980) 1–10.

[26] T. Schrefl, J. Fidler, H. Kronmüller, Remanence and coercivity in isotropic nanocrystalline permanent magnets, Phys. Rev. B 49 (9) (1994) 6100.
[27] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[28] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets, 2021, arXiv preprint arXiv:2103.10974.
[29] A. Kovacs, L. Exl, A. Kornell, J. Fischbacher, M. Hovorka, M. Gusenbauer, L. Breth, H. Oezelt, M. Yano, N. Sakuma, et al., Conditional physics informed neural networks, Commun. Nonlinear Sci. Numer. Simul. 104 (2022) 106041.
[30] W.F. Brown Jr., Some magnetostatic and micromagnetic properties of the infinite rectangular bar, J. Appl. Phys. 35 (7) (1964) 2102–2106.
[31] Q. Chen, A. Konrad, A review of finite element open boundary techniques for static and quasi-static electromagnetic field problems, IEEE Trans. Magn. 33 (1) (1997) 663–676.
[32] H. Kronmüller, Theory of nucleation fields in inhomogeneous ferromagnets, Phys. Status Solidi (B) 144 (1) (1987) 385–396.
[33] D.S. Kinderlehrer, L. Ma, Simulation of hysteresis in nonlinear systems, in: Smart Structures and Materials 1994: Mathematics and Control in Smart Structures, Vol. 2192, International Society for Optics and Photonics, 1994, pp. 78–87.
[34] S.A. Niaki, E. Haghighat, X. Li, T. Campbell, R. Vaziri, Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture, 2020, arXiv preprint arXiv:2011.13511.
[35] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, Vol. 1, MIT press, Cambridge, 2016.
[36] I.M. Sobol, Uniformly distributed sequences with an additional uniform property, USSR Comput. Math. Math. Phys. 16 (5) (1976) 236–242.
[37] E. Haghighat, R. Juanes, Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113552.
[38] R.E. Caflisch, et al., Monte carlo and quasi-monte carlo methods, Acta Numer. 1998 (1998) 1–49.
[39] T. Head, G.L. MechCoder, et al., Scikit-optimize/scikit-optimize: v0. 5.2. 2018, 2018, http://dx.doi.org/10.5281/Zenodo.1207017, 1207017.
[40] Y. Shin, J. Darbon, G.E. Karniadakis, On the convergence and generalization of physics informed neural networks, ArXiv E-Prints (2020) arXiv–2004.
[41] S. Mishra, R. Molinaro, Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs, 2020, arXiv preprint arXiv:2006.16144.
[42] W.F. Brown Jr., A. Morrish, Effect of a cavity on a single-domain magnetic particle, Phys. Rev. 105 (4) (1957) 1198.
[43] A. Aharoni, Demagnetizing factors for rectangular ferromagnetic prisms, J. Appl. Phys. 83 (6) (1998) 3432–3434.
[44] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[45] G. Hinton, N. Srivastava, K. Swersk, Neural networks for machine learning lecture 6, 2012, URL: https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf.
[46] S.M. Assad, C.C. Lim, Circular discrepancy and a Monte Carlo algorithm for generating a low circular discrepancy sequence, in: Vortex Dominated Flows: A Volume Celebrating Lu Ting's 80th Birthday, World Scientific, 2005, pp. 1–19.
[47] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Tang, On large-batch training for deep learning: Generalization gap and sharp minima, 2016, arXiv preprint arXiv:1609.04836.
[48] S. Reddi, M. Zaheer, D. Sachan, S. Kale, S. Kumar, Adaptive methods for non-convex optimization, in: Proceeding of 32nd Conference on Neural Information Processing Systems (NIPS 2018), 2018.
[49] K. Shukla, P.C. Di Leoni, J. Blackshire, D. Sparkman, G.E. Karniadakis, Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, J. Nondestruct. Eval. 39 (3) (2020) 1–20.
[50] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.
[51] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, Deepxde: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[52] M.E. Schabes, H.N. Bertram, Magnetization processes in ferromagnetic cubes, J. Appl. Phys. 64 (3) (1988) 1347–1357.
[53] M.E. Schabes, Micromagnetic theory of non-uniform magnetization processes in magnetic recording particles, J. Magn. Magn. Mater. 95 (3) (1991) 249–288.
[54] F. Chollet, et al., Deep Learning with Python, Vol. 361, Manning New York, 2018.
[55] M. Staib, S. Reddi, S. Kale, S. Kumar, S. Sra, Escaping saddle points with adaptive gradient methods, in: International Conference on Machine Learning, PMLR, 2019, pp. 5956–5965.
[56] S. Hock, Züchtung und magnetische Eigenschaften von (Fe, Al) 14 (Nd, Dy) 2 B-Einkristallen (Ph.D. thesis), Max-Planck-Institut für Metallforschung, Institut für Physik, 1988.
[57] H. Kronmüller, K.-D. Durst, G. Martinek, Angular dependence of the coercive field in sintered Fe77Nd15B8 magnets, J. Magn. Magn. Mater. 69 (2) (1987) 149–157.