Mathematical and Computational Applications

MDPI

Article

An Artificial Neural Network Based Solution Scheme for Periodic Computational Homogenization of Electrostatic Problems

Felix Selim Göküzüm *, Lu Trong Khiem Nguyen and Marc-André Keip

Institute of Applied Mechanics, Chair of Materials Theory, University of Stuttgart, Pfaffenwaldring 7, 70569 Stuttgart, Germany; nguyen@mechbau.uni-stuttgart.de (L.T.K.N.); keip@mechbau.uni-stuttgart.de (M.-A.K.)

* Correspondence: goekuezuem@mechbau.uni-stuttgart.de

Received: 22 March 2019; Accepted: 9 April 2019; Published: 17 April 2019

Abstract: The present work addresses a solution algorithm for homogenization problems based on an artificial neural network (ANN) discretization. The core idea is the construction of trial functions through ANNs that fulfill a priori the periodic boundary conditions of the microscopic problem. A global potential serves as an objective function, which by construction of the trial function can be optimized without constraints. The aim of the new approach is to reduce the number of unknowns as ANNs are able to fit complicated functions with a relatively small number of internal parameters. We investigate the viability of the scheme on the basis of one-, two- and three-dimensional microstructure problems. Further, global and piecewise-defined approaches for constructing the trial function are discussed and compared to finite element (FE) and fast Fourier transform (FFT) based simulations.

Keywords: machine learning; artificial neural networks; computational homogenization

1. Introduction

Artificial neural networks (ANNs) have attracted a lot of attention in the last few years due to their excellent universal approximation properties. Originally developed to model nervous activity in living brains [1,2], they nowadays grow popular in data-driven approaches. Tasks such as image and speech recognition [3,4] or the prediction of users' behavior on social and commercial websites are characterized by a large amount of accessible data compared to a difficult analytic mathematical description. The use of ANNs or other *machine learning algorithms* such as *anomaly detection* [5] and *support vector machines* [6] is suited for such problems as it enables the fitting of even highly complex data with high accuracy. Recent trends in machine learning concern the physical constraining of data driven methods for even higher convergence rate and accuracy, as done in [7].

Due to the aforementioned advantages and improvements, machine learning algorithms gained entry into the field of continuum mechanics and material modeling as well. Successful implementations were performed for the prediction of material response based on the fitting of experimental data through ANNs [8–10]. Another interesting application is the reduction of microstructure data of a given material through pattern recognition in order to reduce computational demands (see, e.g., [11,12]).

In the present work, we employ ANNs to seek the solution of *homogenization problems*. Homogenization aims for the prediction of the macroscopic response of materials that have microstructures described on length scales far lower than the macroscopic dimension. In terms of *analytical* homogenization, Voigt [13] and Reuss [14] were the first to provide *bounds* of effective properties [15]. Hashin and Shtrikman [16] and Willis [17] improved the theory in terms of tighter bounds. Further estimates were developed using the self-consistent method [18,19] and the Mori–Tanaka method [20] afterwards.

In the case of a rather fine microstructures with complex topology and non-linear material behavior, those bounds are only able to make rough predictions on the effective properties. To describe microscopic fields and effective properties in a more detailed and accurate fashion, several *computational* approaches have been developed in the last decades. Two of the most commonly used discretization techniques are finite element (FE) methods [21,22] and fast Fourier transform (FFT) based solvers [23,24]. However, for materials with a microstructure that need fine discretization, the memory cost and solution time of the solvers increases vastly, making multiscale simulations uneconomical. Promising approaches to mitigate these problems are model order reduction methods (see, e.g., [25,26]).

In the present work, a memory efficient solution scheme based on the discretization through ANNs is presented. We therefore follow the idea of Lagaris et al. [27], who introduced a method for solving differential equations through ANN-based trial functions. The functions are constructed in a way that they a priori fulfill the given boundary conditions and the squared error residual of the differential equation is used as an objective that needs to be optimized with respect to the ANNs' weights. The construction of the trial function might be a difficult task for complicated boundary value problems. However, in conventional homogenization problems, we usually deal with rather simple boundary geometries. In the present work, we consider square representative volume elements (RVE) under periodic boundary conditions, as described in Section 2. In Section 3.1, the concept of the ANN-based trial function according to Lagaris et al. [27] is explained. In contrast to Lagaris et al. [27], the optimization objective in our problem is not the squared error residual of a differential equation but emerges from a global energy potential. Sections 3.2 and 3.3 give the ANNs' structure used in the present work as well as the derivatives necessary to optimize the objective function. Finally, in Section 4, the robustness of the presented method is validated for one-, two- and three-dimensional problems and is compared to FE- and FFT-based computations. Further, we compare a global with a piecewise-defined approach for constructing the trial function. In the global approach, the solution is represented by only one global function using several ANNs and the topology of the microstructure must be learned by the neural networks itself. On the other hand, in a piecewise-defined approach, the solution is represented by many neural networks that are piecewise defined on different sub-domains of the RVE. A conclusion is provided in Section 5.

2. Homogenization Framework

In the present work, we consider first-order homogenization of electrostatic problems. The fundamental laws of electrostatics and the corresponding variational formulation are given. In terms of the homogenization framework, we assume separation of length scales between macro- and microscale in the sense that the length scale at which the material properties vary quickly, namely the microscale, is much smaller than the length scale at the same order of the body's dimensions. A connection for the micro- and macrofields is given by the celebrated Hill–Mandel condition [28], which allows for the derivation of consistent boundary conditions for the microscopic boundary value problem.

2.1. Energy Formulation for Electrostatic Problems

We now recall the fundamental equations of electrostatics in the presence of matter [29]. The focus lies on a variational formulation as the energetic potential is later needed in the optimization principle. Assuming that there are neither free currents nor charges, the fundamental physics of electric fields in a body $\mathcal{B}$ are governed by the reduced Maxwell equations

$$\operatorname{curl} \boldsymbol{E} = 0 \quad \text{and} \quad \operatorname{div} \boldsymbol{D} = 0 \quad \text{in} \quad \mathcal{B}, \tag{1}$$

where $\boldsymbol{E}$ denotes the electric field and $\boldsymbol{D}$ the electric displacement. In vacuum, the electric field and displacement are connected through the vacuum permittivity $\kappa_0 \approx 8.854 \cdot 10^{-12} \frac{\mathrm{As}}{\mathrm{Vm}}$ as $\boldsymbol{D} = \kappa_0 \boldsymbol{E}$. In the presence of matter, the permittivity $\kappa = \kappa_0 \cdot \kappa_r$ must be adapted accordingly. To solve Equation (1),

we choose the electric field as our primary variable and construct it as the negative gradient of some scalar electric potential ϕ according to

$$\boldsymbol{E} := -\operatorname{grad}\phi \quad \Rightarrow \quad \operatorname{curl}\boldsymbol{E} \equiv 0, \tag{2}$$

and thus Equation $(1)_1$ is automatically fulfilled. Next, we want to solve Equation $(1)_2$ under some boundary conditions. We therefore introduce an energy potential

$$\Pi(\phi) = \int_{\mathcal{B}} \Psi(\boldsymbol{E})\, dV + \int_{\partial\mathcal{B}_q} q\,\phi\, dA \quad \text{and} \quad \phi = \phi^* \ \text{ on } \ \partial\mathcal{B}_\phi, \tag{3}$$

where $\Psi(\boldsymbol{E})$ is a constitutive energy density, $\partial\mathcal{B}_q$ and $\partial\mathcal{B}_\phi$ denote boundaries along with electric charges q and electric potential ϕ^* as prescribed. From the latter potential, it can be shown that, for equilibrium, i.e. $\delta\Pi = 0$, Equation $(1)_2$ is solved in a weak sense

$$\delta\Pi(\phi) = -\int_{\mathcal{B}} (\operatorname{div}\boldsymbol{D})\delta\phi\, dV + \int_{\partial\mathcal{B}_q} (q + \boldsymbol{D}\cdot\boldsymbol{N})\,\delta\phi\, dA = 0 \quad \text{with} \quad \boldsymbol{D} = -\frac{\partial\Psi}{\partial\boldsymbol{E}}, \tag{4}$$

for all $\delta\phi$. Here, $\boldsymbol{N}$ denotes the unit normal vector pointing outwards of $\partial\mathcal{B}$. By choosing $\Psi = -1/2\kappa\,\boldsymbol{E}\cdot\boldsymbol{E}$, the static Maxwell equations are recovered.

2.2. Microscopic Boundary Value Problem

In the present work, we consider homogenization problems governed by the existence of so-called representative volume elements (RVEs). They are chosen in a way that they are statistically representative for the overall microstructure of the material. Fields emerging on the microscale are driven by macroscopic fields, which are assumed to be constant in the RVE due to the separation of length scales. The separation of length scales leads to a degeneration of the RVEs to points on the macroscale. Scale transition rules can be derived from the Hill–Mandel condition [28] by postulating energy conservation between the microscopic RVE and a macroscopic observation in the form

$$\overline{\Pi} = \sup_{\boldsymbol{E}} \frac{1}{|\mathcal{B}|}\Pi(\phi), \tag{5}$$

where the macroscopic energy potential $\overline{\Pi}$ is obtained at equilibrium of the microscopic energy potential $\Pi(\phi)$. According to Equation (3), the internal energy density appearing in the potential is now a function of the electric field. In line with the assumption of first-order homogenization and separation of length scales, the electric field vector

$$\boldsymbol{E} = \overline{\boldsymbol{E}} - \nabla\widetilde{\phi} \tag{6}$$

is decomposed into a macroscopic contribution

$$\overline{\boldsymbol{E}} = \frac{1}{|\mathcal{B}|}\int_{\mathcal{B}} \boldsymbol{E}\, \mathrm{d}V, \tag{7}$$

which is constant on the microscale, and the gradient of the fluctuative scalar electric potential $\widetilde{\phi}$ that acts as primary variable. The system is closed by applying appropriate boundary conditions on the RVE. There are several boundary conditions that fulfill the Hill–Mandel condition (5). In the present work, we focus on periodic boundary conditions of the form

$$\widetilde{\phi}(\boldsymbol{x}^+) = \widetilde{\phi}(\boldsymbol{x}^-) \quad \text{and} \quad \boldsymbol{D}\cdot\boldsymbol{N}(\boldsymbol{x}^+) = -\boldsymbol{D}\cdot\boldsymbol{N}(\boldsymbol{x}^-) \quad \text{on} \quad \partial\mathcal{B}, \tag{8}$$

where $\boldsymbol{x}^{\pm}$ indicate opposite coordinates at the boundary of the RVE [30–32]. Note that we only need to prescribe one of the two boundary conditions given in the latter equation as the other one will emerge naturally in equilibrium.

3. Artificial Neural Network Based Solution Scheme

In this section, a solution procedure based on *artificial neural networks* (ANNs) for finding the equilibrium state of the potential in Equation (3) is presented. Core idea is the construction of trial functions, which consist of ANNs and multiplicative factors fulfilling the given set of boundary conditions in Equation (8). We then recapitulate the two main ANN structures used in the present work, namely the single layer perceptron net and the multilayer perceptron. Additionally, the derivatives of those nets with respect to its inputs as well as with respect to its weights are given as they are needed when using gradient descent methods.

3.1. Optimization Principle

Consider the previously introduced RVE occupying the space $\mathcal{B}$. Our goal is to find the microscopic equilibrium state of a given global potential for this body. Under prescribed periodic Dirichlet boundary conditions in Equation $(8)_1$, the potential in Equation (3) takes the form

$$\Pi(\phi) = \int_{\mathcal{B}} \Psi(\boldsymbol{x}, \boldsymbol{E})\, dV. \tag{9}$$

Having the physical problem covered, the question arises how to approximate the solution fields. While finite element approaches employ local shape functions and Fourier transform based methods use global trigonometric basis functions, in the present work, we want to investigate an approximation method based on artificial neural networks. Following the idea of Lagaris et al. [27], we construct a trial function ϕ_t using arbitrary artificial neural networks $N_i(\boldsymbol{x}, \boldsymbol{p}_i)$ as

$$\widetilde{\phi}_t(\boldsymbol{x}, \boldsymbol{p}) = A_0(\boldsymbol{x}) + A_1(\boldsymbol{x})N_1(\boldsymbol{x}, \boldsymbol{p}_1) + A_2(\boldsymbol{x})N_2(\boldsymbol{x}, \boldsymbol{p}_2) + ... + A_n(\boldsymbol{x})N_n(\boldsymbol{x}, \boldsymbol{p}_n), \tag{10}$$

where $A_i(\boldsymbol{x})$ are functions ensuring that the boundary conditions are fulfilled a priori. As a generic one-dimensional example, one could think of a scalar electric potential that should fulfill the boundary conditions $\widetilde{\phi}(0) = \widetilde{\phi}(1) = 1$. In this case, we would have $A_0 = 1$ and $A_1 = x(1-x)$, yielding the trial function according to Equation (10) as $\widetilde{\phi}_t(x, \boldsymbol{p}) = 1 + x(1-x)N_1(x, \boldsymbol{p}_1)$. The corresponding electric field $\boldsymbol{E}_t(\boldsymbol{x}, \boldsymbol{p})$ in line with Equation (6) can be calculated analytically according to the neural network derivatives given in Sections 3.2 and 3.3. Using the gradient field along with Equation (9) gives the global potential in terms of the neural network's parameters as follows

$$\Pi(\boldsymbol{p}) = \int_{\mathcal{B}} \Psi(\boldsymbol{x}, \boldsymbol{E}_t(\boldsymbol{x}, \boldsymbol{p}))\, dV. \tag{11}$$

Finally, the objective function in our machine learning problem is obtained from the Hill–Mandel condition in Equation (5) in combination with the global potential in Equation (11) approximated by neural networks. It appears as

$$\boxed{\overline{\Pi} = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|} \Pi(\boldsymbol{p}),} \tag{12}$$

where the optimization is carried out with respect to the neural network's parameters. By having the periodic boundary conditions fulfilled by construction, the optimization can be carried out without any constraints on the parameters $\boldsymbol{p}$.

3.2. Single Layer Perceptron (SLP)

The *single layer perceptron* (SLP) is one of the most basic versions of artificial neural networks [33]. It is a reduced version of the general structure depicted in Figure 1. An SLP only consists of one hidden layer that connects the input and the output. Assuming an input vector $\boldsymbol{x}$ of dimension d, the response N of an SLP is calculated as

$$N(\boldsymbol{x}, \boldsymbol{p}) = \sum_{i=1}^{H} v_i \sigma(z_i) + \bar{b} \quad \text{with} \quad z_i = \sum_{j=1}^{d} w_{ij} x_j + u_i, \tag{13}$$

where v_i, w_{ij}, u_i and $\bar{b}$ are the weights and biases of the hidden unit and the output bias, respectively, and H is the overall number of neurons in the hidden layer. Those weights and biases are assembled in the neural network parameter vector $\boldsymbol{p}$. Here, σ denotes an activation function of a neuron. The activation functions may be chosen problem-dependent and can have a large impact on the training behavior of the artificial neural network. Figure 2 shows the three activation functions used in the present work, namely the logistic sigmoid, the hyperbolic tangent and the softplus function. Despite its popularity in machine learning tasks, we are not using the rectifier linear unit (ReLu) activation function in the present work. First tests using the ReLu function resulted in poor convergence rates. We suspect that this stems from errors in the numerical integration when using the ReLu function and its discontinuous derivative. The derivatives of the SLP net with respect to its input then appear as

$$N_{,j} = \frac{\partial N}{\partial x_j} = \sum_{i=1}^{H} v_i w_{ij} \sigma'(z_i), \tag{14}$$

where $\sigma'(z_i)$ denotes the derivative of the sigmoid function with respect to its argument. The spatial derivative can be perceived as an SLP with modified weights and activation function but it is now a gradient field. To use efficient gradient based solvers when optimizing the weights of the neural network, it is convenient to have the explicit derivatives of the ANNs with respect to the weights. These can be obtained as

$$\begin{array}{ll} \dfrac{\partial N}{\partial u_i} = v_i \sigma'(z_i), & \dfrac{\partial N_{,j}}{\partial u_i} = v_i w_{ij} \sigma''(z_i), \\ \dfrac{\partial N}{\partial w_{ij}} = v_i x_j \sigma'(z_i), & \dfrac{\partial N_{,j}}{\partial w_{im}} = x_m v_i w_{ij} \sigma''(z_i) + v_i \sigma'(z_i) \delta_{jm}, \\ \dfrac{\partial N}{\partial \bar{b}} = 1, & \dfrac{\partial N_{,j}}{\partial \bar{b}} = 0, \\ \dfrac{\partial N}{\partial v_i} = \sigma(z_i), & \dfrac{\partial N_{,j}}{\partial v_i} = w_{ij} \sigma'(z_i). \end{array} \tag{15}$$

Note that, in the derivatives above, the indices are *not* treated by Einstein's summation convention. Indices appearing twice are rather multiplied pointwise in MATLAB [34] convention.

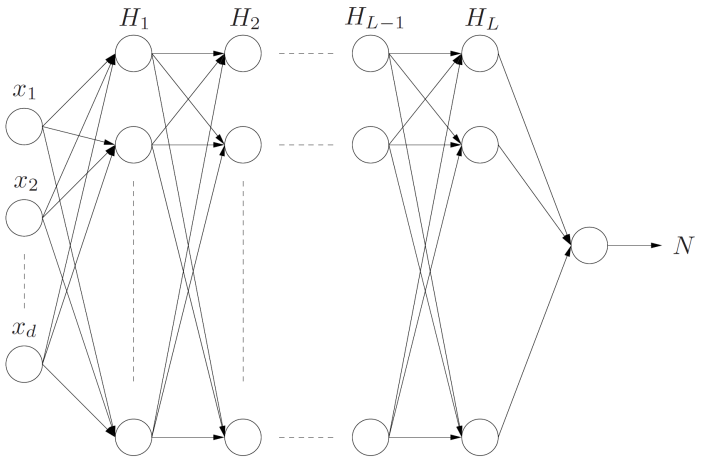

Figure 1. General structure of a multilayer perceptron with input $\boldsymbol{x}$, L hidden layers and output N. Each neuron evaluates its inputs through predefined activation functions.

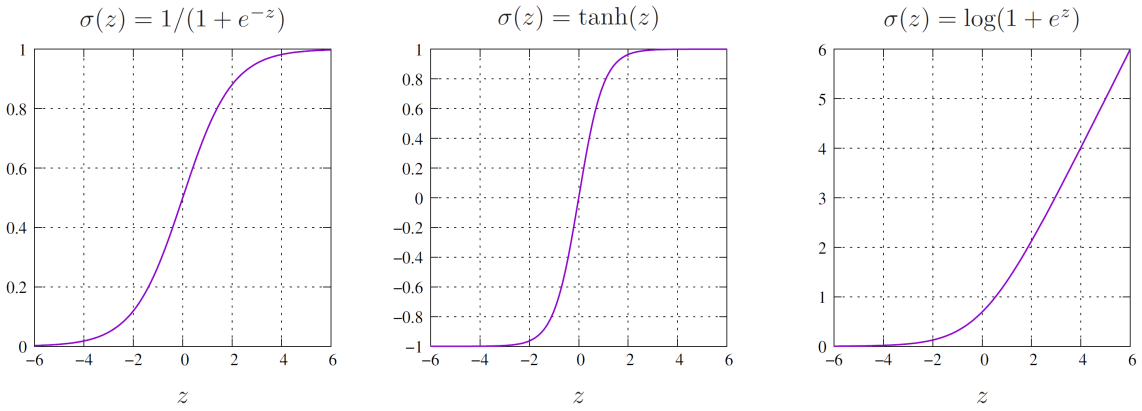

Figure 2. Different types of popular activation functions: logistic sigmoid, hyperbolic tangent and softplus function.

3.3. Multilayer Perceptron (MLP)

The multilayer perceptron (MLP) works similarly to the single layer perceptron. However, it is constructed by a higher number L of hidden layers. It can be shown that this deep structure enables a more general approximation property of the neural network and might lead to better training behavior [35]. In the present work, we focus on MLPs with only two hidden layers. The output is then computed as

$$N(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=1}^{H_2} v_k \sigma(r_k) + \overline{b}, \quad r_k = \sum_{i=1}^{H_1} \theta_{ki} \sigma(z_i) + c_k \quad \text{with} \quad z_i = \sum_{j=1}^{d} w_{ij} x_j + u_i, \tag{16}$$

where we have now additional weights θ_{ki} and biases c_k associated with the second hidden layer. The spatial derivative of the MLP appears as

$$N_{,j} = \frac{\partial N}{\partial x_j} = \sum_{k=1}^{H_2} \sum_{i=1}^{H_1} v_k \sigma'(r_k) \theta_{ki} \sigma'(z_i) w_{ij}. \tag{17}$$

The derivatives of the MLP with respect to the weights are computed as

$$
\begin{aligned}
&\frac{\partial N}{\partial u_i} = \sum_{k=1}^{H_2} v_k \sigma'(r_k)\theta_{ki}\sigma'(z_i), && \frac{\partial N_{,j}}{\partial u_i} = \sum_{k=1}^{H_2} \Big[v_k \sigma''(r_k)\theta_{ki}\sigma'(z_i) \sum_{i=1}^{H_1} [w_{ij}\theta_{ki}\sigma'(z_i)] \\
& && \quad + v_k \sigma'(r_k)\theta_{ki}\sigma''(z_i) w_{ij}) \Big], \\
&\frac{\partial N}{\partial w_{ij}} = \sum_{k=1}^{H_2} v_k \sigma'(r_k)\theta_{ki}\sigma'(z_i) x_j, && \frac{\partial N_{,j}}{\partial w_{im}} = \sum_{k=1}^{H_2} \Big[v_k \sigma''(r_k)\theta_{ki}\sigma'(z_i) x_m \sum_{i=1}^{H_1} [w_{ij}\theta_{ki}\sigma'(z_i)] \\
& && \quad + v_k \sigma'(r_k)\theta_{ki}\sigma''(z_i) w_{ij} x_m \\
& && \quad + v_k \sigma'(r_k)\theta_{ki}\sigma'(z_i)\delta_{im} \Big], \\
&\frac{\partial N}{\partial c_k} = v_k \sigma'(r_k), && \frac{\partial N_{,j}}{\partial c_k} = v_k \sigma''(r_k) \sum_{i=1}^{H_1} [w_{ij}\theta_{ki}\sigma'(z_i)], \\
&\frac{\partial N}{\partial \theta_{ki}} = v_k \sigma'(r_k)\sigma(z_i), && \frac{\partial N_{,j}}{\partial \theta_{ki}} = v_k \sigma''(r_k)\sigma'(z_i) \sum_{i=1}^{H_1} [w_{ij}\theta_{ki}\sigma(z_i)] \\
& && \quad + v_k \sigma'(r_k)\sigma'(z_i) w_{ij}, \\
&\frac{\partial N}{\partial \overline{b}} = 1, && \frac{\partial N_{,j}}{\partial \overline{b}} = 0, \\
&\frac{\partial N}{\partial v_k} = \sigma(r_k), && \frac{\partial N_{,j}}{\partial v_k} = \sigma'(r_k) \sum_{i=1}^{H_1} [w_{ij}\theta_{ki}\sigma'(z_i)].
\end{aligned}
\tag{18}
$$

4. Numerical Examples

In this section, we test the robustness and reliability of the proposed method on a set of one-, two- and three-dimensional problems. The influence of global versus piecewise-defined constructions of the trial functions as well as the impact of the neuron count on the simulation results are explored. The one-dimensional example is implemented in MATLAB [34] while the two- and three-dimensional examples are carried out by a Fortran 77 code that utilizes the L-BFGS-B optimization algorithms [36,37]. For the sake of simplicity, all the following simulations are performed with normalized units.

4.1. One-Dimensional Example

We first consider a one-dimensional problem to demonstrate the features of the proposed method. The RVE is a simple two-phase laminate of unit length $l = 1$ and unit cross section, which is loaded with a macroscopic electric field $\overline{E} = 0.01$. The global potential takes the form

$$
\Pi(\phi) = \int_{\mathcal{B}} \Psi(E, x)\, dx = -\int_{\mathcal{B}} \frac{1}{2} \kappa E^2\, dx, \tag{19}
$$

where $\kappa_1 = 1$ for $x < 0.5$ and $\kappa_2 = 2$ for $x > 0.5$. Having the decomposition in Equation (6)

$$
E = \overline{E} - \frac{\partial \widetilde{\phi}}{\partial x} \tag{20}
$$

along with the boundary conditions $\widetilde{\phi}(0) = 0$ and $\widetilde{\phi}(1) = 0$, the analytical solution for the electric field reads

$$
E = \begin{cases} \dfrac{2\kappa_2}{\kappa_1 + \kappa_2}\overline{E} & 0 \le x < l/2 \\ \dfrac{2\kappa_1}{\kappa_1 + \kappa_2}\overline{E} & l/2 < x \le l \end{cases} \tag{21}
$$

and for the electric potential

$$\widetilde{\phi} = \begin{cases} \overline{E}\dfrac{\kappa_1 - \kappa_2}{\kappa_1 + \kappa_2} x & 0 \leq x \leq l/2 \\ \overline{E}\dfrac{\kappa_2 - \kappa_1}{\kappa_1 + \kappa_2}(x - l) & l/2 \leq x \leq l \end{cases}. \tag{22}$$

4.1.1. Global Neural Net Approach on Equidistant Grid

To find the electric scalar potential that optimizes the energy potential in Equation (19), we construct a global trial function according to Equation (10) that automatically fulfills the boundary conditions given above as

$$\widetilde{\phi}_t = A_o + A_1(x)N(x, \boldsymbol{p}) = x(1 - x)N(x, \boldsymbol{p}), \tag{23}$$

where N is a neural network that takes x as an input and has the weights and biases $\boldsymbol{p}$. The derivatives in this case can be computed explicitly as

$$\frac{\partial \widetilde{\phi}_t}{\partial x} = (1 - 2x)N(x, \boldsymbol{p}) + x(1 - x)\frac{\partial N(x, \boldsymbol{p})}{\partial x}. \tag{24}$$

The derivatives then allow us to compute the global potential in terms of the neural network's weights and biases as

$$\Pi(\boldsymbol{p}) = -\int_{\mathcal{B}} \frac{1}{2}\kappa(\overline{E} - \frac{\widetilde{\phi}_t}{\partial x})^2 \, dx. \tag{25}$$

Finally, we need a numerical integration scheme to evaluate the integral. In the present work, we use quadrature points in terms of equidistant grid points x_k with distance Δx on the interval of $[0, 1]$ as $\{\Delta x/2, 3\Delta x/2, \ldots, 1 - \Delta x/2\}$, yielding the discrete objective

$$\boxed{\overline{\Pi} = \sup_{\boldsymbol{p}} \frac{1}{l} \sum_{0 < x_k < 1.0} -\frac{1}{2}\kappa(\overline{E} - \frac{\widetilde{\phi}_t}{\partial x})^2 \Delta x.} \tag{26}$$

The maximum of this objective function can be found by means of the gradient descent method. The gradients of the objective with respect to the weights $\boldsymbol{p}$ that are needed for such an iterative solver can be computed using Equation (15). We then have everything at hand to carry out the first numerical example. As for the ANN architecture, we use an SLP with $H = 10$ neurons in the hidden layer. As for the activation function, the logistic sigmoid function $\sigma(z) = 1/(1 + e^{-z})$ is chosen, and the weights and biases $\boldsymbol{p}$ are randomly initalized with a uniform distribution between 0 and 1. Figure 3 shows the result of the numerical experiment (green) compared to the analytical results (black). One can see that the numerical scalar electric potential $\widetilde{\phi}_t$ is close to the analytical solution. However, having a look at the gradients in the form of E reveals the occurrence of oscillations around the discontinuity. The MATLAB [34] code that generates these results can be found in Appendix A. Note that the tolerances for the step size and the optimality as well as the maximum number of function evaluations and iterations is set different from the MATLAB [34] default values to obtain reasonable results. One might decrease the oscillations by having even higher iteration numbers. However, the jump in the solution at the vicinity of the material jump cannot be captured by the global smooth neural network function. The more neurons we use, the better accuracy we obtain, which leads to a trade-off between accuracy and speed.

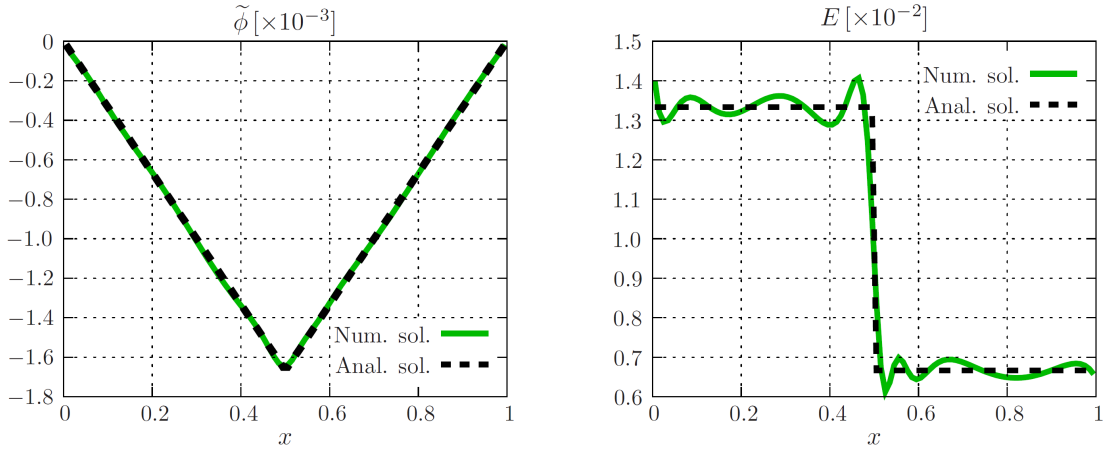

Figure 3. Numerical vs. analytical solution for a global trial function approach using an SLPs with 10 neurons and a random initialization of $\boldsymbol{p}^{(0)} \sim \boldsymbol{\mathcal{U}}(0,1)$, where $\boldsymbol{\mathcal{U}}(0,1)$ denotes a vector of numbers generated from a uniform distribution between 0 and 1. These parameters are statistically independent.

4.1.2. Piecewise-defined Neural Net Approach on Equidistant Grid

To overcome the oscillations observed in the global approach, we next construct a trial function that is piecewise defined for the RVE's different material regions

$$\widetilde{\phi}_t = \begin{cases} xN_1(x,\boldsymbol{p}_1) & 0 < x \leq 0.5 \\ (0.5-x)(1-x)N_2(x,\boldsymbol{p}_2) + 2(1-x)\widetilde{\phi}_t(0.5) & 0.5 < x \leq 1.0 \end{cases}. \tag{27}$$

Its derivatives can be computed according to

$$\frac{\partial \widetilde{\phi}_t}{\partial x} = \begin{cases} N_1(x,\boldsymbol{p}_1) + x\dfrac{\partial N_1(x,\boldsymbol{p}_1)}{\partial x} & 0 < x < 0.5 \\ (2x-1.5)N_2(x,\boldsymbol{p}_2) + (0.5-x)(1-x)\dfrac{\partial N_2(x,\boldsymbol{p}_2)}{\partial x} - 2\widetilde{\phi}_t(0.5) & 0.5 < x < 1.0 \end{cases}. \tag{28}$$

The global potential in terms of the neural network's weights and biases then appears as

$$\Pi(\boldsymbol{p}) = \int_{\mathcal{B}} \frac{1}{2}\kappa(\overline{E} - \frac{\widetilde{\phi}_t}{\partial x})^2\, dx. \tag{29}$$

In analogy to the global approach, numerical integration is performed using equidistant grid points x_k with distance Δx as quadrature points in the interval $[0,1]$ to obtain the discrete objective

$$\boxed{\overline{\Pi} = \sup_{p} \frac{1}{l}\left(\sum_{0<x_k<0.5} -\frac{1}{2}\kappa_1(\overline{E} - \frac{\widetilde{\phi}_t}{\partial x})^2 - \sum_{0.5<x_k<1.0} \frac{1}{2}\kappa_2(\overline{E} - \frac{\widetilde{\phi}_t}{\partial x})^2 \right)\Delta x.} \tag{30}$$

The gradient of the objective function with respect to the parameters $\boldsymbol{p}$ can again be obtained through the derivatives of Equation (15). However, the output and training behavior of artificial neural networks is dependent on the initialization of the weights and biases $\boldsymbol{p}$. In a first run, we set the number of neurons in the two SLPs' hidden layers to 10 and initialize the weights randomly between 0 and 1. As for the activation function, we choose the logistic sigmoid function $\sigma(z) = 1/(1+e^{-z})$.

Figure 4 shows the numerical results in green compared to the analytical results in black. There is a distinct deviation between them. Having a look at the output layer of the SLP in Equation (13) and

its derivative in Equation (14), one can see that the initial output for 10 neurons for an initialization of all weights between 0 and 1 is far from the exact solution. This difference becomes even larger for higher number of neurons. In contrast to the global approach, we use the default values of the unconstrained MATLAB [34] minimizer and, apparently, there are too few iterations.

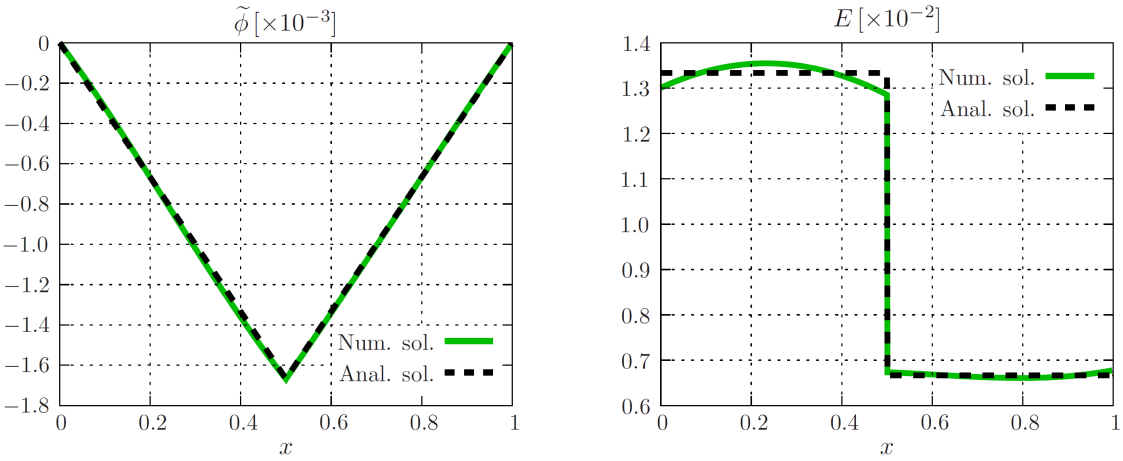

Figure 4. Numerical vs. analytical solution for 2 SLPs with 10 neurons each and a random initialization of $\boldsymbol{p}^{(0)} \sim \boldsymbol{\mathcal{U}}(0,1)$, where $\boldsymbol{\mathcal{U}}(0,1)$ denotes a vector of numbers generated from a uniform distribution between 0 and 1. These parameters are statistically independent.

Next, we want to have fewer iterations of the solver within the default values by using an adaptive way of initializing the weights. The key idea is to initialize the weights in a way that the net and its derivative output values are roughly in the range of the values we would expect with respect to the given macroscopic load. We therefore use a simple modification of the weight initialization given as

$$\boldsymbol{p}^{(0),*} \sim \frac{\overline{E}}{H} * \boldsymbol{\mathcal{U}}(0,1), \tag{31}$$

where $\boldsymbol{\mathcal{U}}(0,1)$ is a vector of random numbers uniformly distributed along 0 and 1. Please note that we use boldface calligraphic $\boldsymbol{\mathcal{U}}$ to indicate the vector notation instead of univariate distribution. The results of the computation using the adaptive initialization method can be seen in Figure 5. The results are closer to the analytical solution and are independent of the number of hidden neurons used, making the overall method much more reliable. The MATLAB [34] code used for generating Figure 5 can be found in Appendix B.

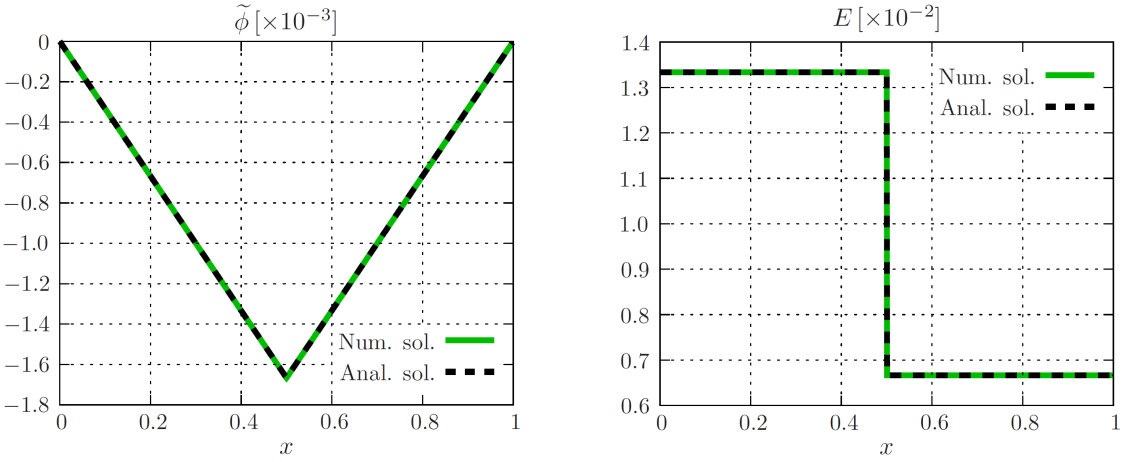

Figure 5. Numerical vs. analytical solution for 2 SLPs with 10 neurons each and a random initialization of $\boldsymbol{p}^{(0),*} \sim \frac{\overline{E}}{H} * \boldsymbol{\mathcal{U}}(0,1)$, where $\boldsymbol{\mathcal{U}}(0,1)$ denotes a vector of numbers generated from a uniform distribution between 0 and 1. These parameters are statistically independent.

4.2. Two-Dimensional Example

Next, we consider a two-dimensional microstructure with a circular inclusion, as shown in Figure 6. The radius of the inclusion is $r_0 = 0.178l$, corresponding to 10% volume fraction. The global energy potential per unit out-of-plane thickness is given as the integral of the internal energy density over the RVE's domain

$$\Pi = \int_{\mathcal{B}} \Psi(\boldsymbol{E})\, dA = -\int_{\mathcal{B}} \frac{1}{2}\kappa \boldsymbol{E}\cdot\boldsymbol{E}\, dA. \tag{32}$$

In this example, the material parameter is set to $\kappa = 1$ in the matrix and $\kappa = 10$ in the inclusion. The square RVE of unit length $l = 1$ is loaded with the constant macroscopic field $\overline{E}_1 = 1$ and $\overline{E}_2 = 0$ under the periodic Dirichlet boundary conditions in Equation $(8)_1$. For reference, a simulation using the finite element method is performed for this optimization problem. The mesh is discretized by linear quadrilateral elements with four Gauss points. The optimized global potential is calculated to $|\overline{\Pi}|^{\text{FEM}} = 0.588652$. Figure 6 shows the contour plot of the microscopic field E_1. The finite element results serve as a reference for the neural network based approaches in the following examples.

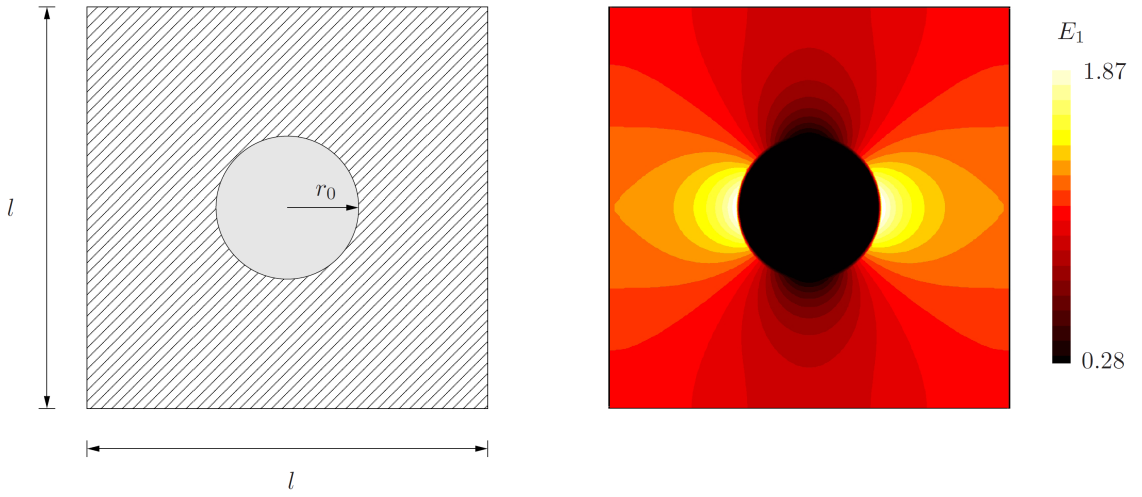

Figure 6. Microstructure of length l with a circular inclusion having the radius $r_0 = 0.178l$. On the right, a finite element solution for a phase contrast of 10 and a loading of $\overline{E}_1 = 1$ and $\overline{E}_2 = 0$ is displayed.

4.2.1. Global Neural Net Approach on Equidistant Grid

First, we want to build a global trial function, which is covering the whole RVE. As we want to implement periodic boundary conditions, a set of three neural networks is used to construct the trial function ϕ_t: $N_{x_1}(x_1)$ acting in the x_1-direction, $N_{x_2}(x_2)$ acting in the x_2-direction and $N_1(\boldsymbol{x})$ acting in both directions. The trial function then takes the form

$$\begin{aligned}\widetilde{\phi}_t(\boldsymbol{x}, \boldsymbol{p}) &= A_1(x_1, x_2)N_1(\boldsymbol{x}, \boldsymbol{p}_1) + A_2(x_1)N_{x_1}(x_1, \boldsymbol{p}_{x_1}) + A_3(x_2)N_{x_2}(x_2, \boldsymbol{p}_{x_2}) \\ &= x_1(1-x_1)x_2(1-x_2)N_1(\boldsymbol{x}, \boldsymbol{p}_1) \\ &\quad + x_1(1-x_1)N_{x_1}(x_1, \boldsymbol{p}_{x_1}) + x_2(1-x_2)N_{x_2}(x_2, \boldsymbol{p}_{x_2}).\end{aligned} \tag{33}$$

Figure 7 shows a visualization of the functions ensuring the boundary conditions. Here, we use SLPs for the boundary networks and a two-layer MLP for the two-dimensional neural network. As for the activation function for the neurons, the hyperbolic tangent $\sigma(z) = \tanh(z)$ is chosen. The negative gradient of the trial function can then be computed to obtain the trial field

$$\boldsymbol{E}_t = \overline{\boldsymbol{E}} - \nabla\widetilde{\phi}_t, \tag{34}$$

where we drop dependencies on $\boldsymbol{x}$ and $\boldsymbol{p}$ in our notation. According to Equation (32), we arrive at the global potential as an objective

$$\overline{\Pi} = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|}\Pi(\boldsymbol{p}) = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|}\int_{\mathcal{B}} -\frac{1}{2}\kappa(\boldsymbol{x})\boldsymbol{E}_t \cdot \boldsymbol{E}_t \, dA. \tag{35}$$

The spatial gradient of the trial function $\nabla\widetilde{\phi}_t$ as well as gradients of the global potential $\partial\Pi(\boldsymbol{p})/\partial\boldsymbol{p}$ used in optimization algorithms can be obtained analytically through the derivatives given in Sections 3.2 and 3.3. We use equidistant grid points for establishing a regular mesh of elements and employ nine Gauss points per element as the quadrature points for numerical integration. The objective is optimized without any constraints on the parameters $\boldsymbol{p}$, as we satisfy the boundary conditions a priori by the construction of $\widetilde{\phi}_t$.

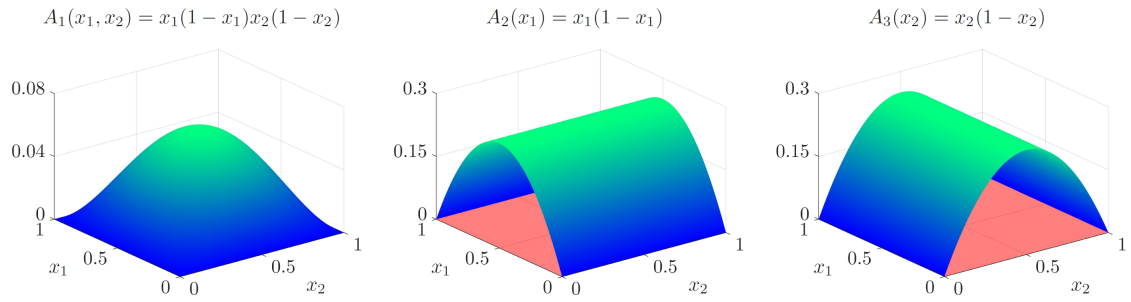

Figure 7. Visualization of the functions A_1, A_2 and A_3 ensuring periodicity of the two-dimensional trial function $\widetilde{\phi}_t$ in a square RVE of unit length $l = 1$. One can see that A_1 covers the volume, A_2 satisfies the periodicity in x_1-direction and A_3 satisfies the periodicity in x_2-direction.

The simulation is carried out for a different number of neurons and integration points as follows: (a) 51×51 elements, 15 neurons in each of the two hidden layers of the MLP and 5 neurons each in the layer of the boundary SLPs; (b) 51×51 elements, 10 neurons in each of the two hidden layers of the MLP and 5 neurons each in the layer of the boundary SLPs; and (c) 101×101 elements, 15 neurons in each of the two hidden layers of the MLP and 5 neurons each in the layer of the boundary SLPs. Additionally, all three set ups are run with a uniform initialization of the weights through

$$\boldsymbol{p}^{(0)} \sim \boldsymbol{\mathcal{U}}(-1, 1). \tag{36}$$

Figure 8 shows the contour plot of E_{t1} of the neural network after 20,000 iterations. One can see that the neural network in Case (a) localizes in an unphysical state. This could be a problem related to overfitting. Case (b), with the same amount of integration points but lower neuron count, shows a qualitatively better result. The overall optimization seems to become more reliable as the integration is performed more accurately, as seen in Case (c). Quantitatively, the global potentials are: (a) $|\overline{\Pi}|(\boldsymbol{p}) = 0.213522$; (b) $|\overline{\Pi}|(\boldsymbol{p}) = 0.590266$; and (c) $|\overline{\Pi}|(\boldsymbol{p}) = 0.589887$ compared to the FEM potential of $|\overline{\Pi}|^{\mathrm{FEM}}(\boldsymbol{E}) = 0.588652$, taken from the simulation that creates Figure 6. In two dimensions and with complicated geometries at hand, it can be difficult to estimate the magnitude of the solution fields appearing a priori. A rather simple approach for initializing the weights as described in the one-dimensional case did not seem to significantly improve the method. Here, we test a second approach of initializing the weights according to a normal distribution

$$\boldsymbol{p}^{(0),*} \sim \boldsymbol{\mathcal{N}}(\mu, \sigma^2), \tag{37}$$

where we set the mean $\mu = 0$ and the variance $\sigma^2 = 1$. Figure 8 shows the contour plot of E_{t1} of the neural network after 20,000 iterations for the normal distribution. Case (a) now shows a more physical state. The energies are calculated as: (a) $|\overline{\Pi}|(\boldsymbol{p}) = 0.590054$; (b) $|\overline{\Pi}|(\boldsymbol{p}) = 0.592981$; and (c) $|\overline{\Pi}|(\boldsymbol{p}) = 0.590544$. At this point, there is surely some potential left for improving weight initialization. This is subject to other fields of machine learning as well, especially in the context of training speed and vanishing gradient problems [38].

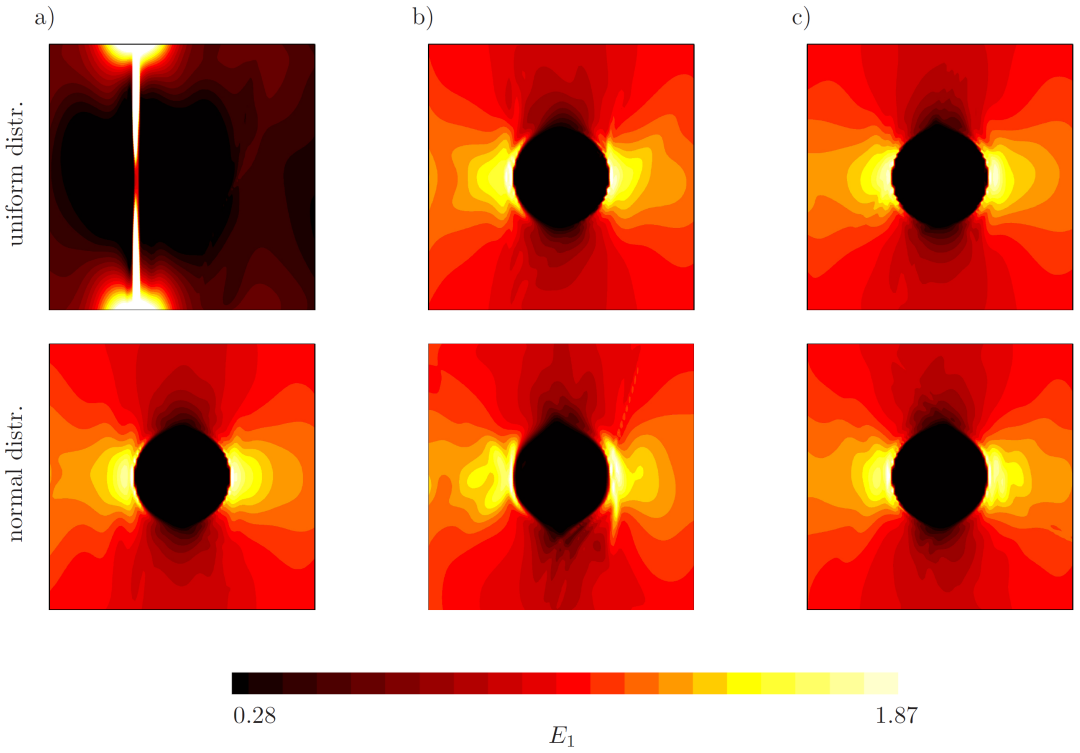

Figure 8. Contour plot of E_{t1} for a set of parameters: (**a**) 51 × 51 elements, 15 neurons per layer in N_1 and 5 neurons each for N_{x_1} and N_{x_2}; (**b**) 51 × 51 elements, 10 neurons per layer in N_1 and 5 neurons each for N_{x_1} and N_{x_2}; and (**c**) 101 × 101 elements, 15 neurons per layer in N_1 and 5 neurons each for N_{x_1} and N_{x_2}.

4.2.2. Piecewise-Defined Neural Net Approach

To further improve the training and approximation properties of the trial functions, we next construct it in a way that it captures the discontinuity of material properties in the microstructure a priori. This is possible due to the simple topology of the microstructure at hand. However, for more complicated microstructures, defining explicit expressions for the trial function might be difficult. We now need a set of four ANNs for the construction of ϕ_t: $N_{x_1}(x_1)$ acting in the x_1-direction, $N_{x_2}(x_2)$ acting in the x_2-direction, $N_1(\boldsymbol{x})$ acting in the matrix and $N_2(\boldsymbol{x})$ acting in the inclusion. The global trial function is defined piecewise in two sub-domains as follows

$$\widetilde{\phi}_t = \begin{cases} x_1(1-x_1)x_2(1-x_2)N_1(\boldsymbol{x},\boldsymbol{p}_1) + x_1(1-x_1)N_{x_1}(x_1,\boldsymbol{p}_{x_1}) & r \geq r_0 \\ +x_2(1-x_2)N_{x_2}(x_2,\boldsymbol{p}_{x_2}) & \\ \widetilde{\phi}_t(r_0,\varphi) + (r_0 - r)N_2(\boldsymbol{r},\boldsymbol{p}_1) & r < r_0 \end{cases}. \tag{38}$$

One can see that the above equations automatically fulfill periodicity at the boundaries as well as the transition condition at the phase interface of the circular inclusion. Here, we implement the trial function in Cartesian coordinates for the matrix and in polar coordinates for the inclusion. Thus, the neural network of the inclusion takes the coordinate vector $\boldsymbol{r} = (r, \varphi)$ in terms of the radius r and the angle φ as its input. Having the origin of our coordinate system in the bottom left corner, the transformation used here takes the form

$$x_1 = 0.5 + r\cos\varphi \quad \text{and} \quad x_2 = 0.5 + r\sin\varphi. \tag{39}$$

With the coordinate transformation at hand, one can calculate the gradient in the matrix and inclusion as

$$\boldsymbol{E}_t = -\nabla(-\overline{\boldsymbol{E}} \cdot \boldsymbol{x} + \widetilde{\phi}_t), \tag{40}$$

where $\overline{\boldsymbol{E}}$ is the applied macroscopic electric field. For a more detailed derivation, see Appendix C. Finally, the potential to optimize in this example takes the form

$$\overline{\Pi} = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|}\Pi(\boldsymbol{p}) = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|}\left(-\int_{\mathcal{B}^{\text{matr}}} \frac{1}{2}\kappa_1 \boldsymbol{E}_t \cdot \boldsymbol{E}_t \, dA - \int_{\mathcal{B}^{\text{incl}}} \frac{1}{2}\kappa_2 \boldsymbol{E}_t \cdot \boldsymbol{E}_t \, dA\right). \tag{41}$$

As a numerical integration scheme, we use a simple one-point quadrature rule, where we have an equidistant grid of 3240 integration points in the matrix and 3600 integration points in the inclusion. The MLP $N_1(\boldsymbol{x}, \boldsymbol{p}_1)$ has two layers with five neurons in each, the SLP $N_2(\boldsymbol{x}, \boldsymbol{p}_2)$ has one layer with four neurons and the boundary SLPs $N_{x_1}(x_1, \boldsymbol{p}_{x_1})$ and $N_{x_2}(x_2, \boldsymbol{p}_{x_2})$ have seven neurons in the hidden layer. This sums up to a total of 116 degrees of freedom $\boldsymbol{p}$. In the present example, we use the hyperbolic tangent $\sigma(z) = \tanh(z)$ as the activation function of the neurons. The initial weights are randomly initialized with uniform distribution in the range of -1 to 1. As in the previous examples, the applied macroscopic load in Equation (40) is $\overline{E}_1 = 1.0$ and $\overline{E}_2 = 0.0$.

Figure 9 shows the result for E_{t1} after 10,000 iterations for the local construction of the trial function. Compared with the previous simulations using only one global trial function, the fields have fewer oscillations. Additionally, the maximum energy after 10,000 iterations is $|\overline{\Pi}|(\boldsymbol{p}) = 0.588414$, being lower than in the previous simulations and lower than the maximum energy computed with finite elements. Interestingly, in the first iterations, the learning rates are higher for the global schemes, whether the weights are initialized according to a normal distribution or a uniform distribution (see Figure 9).

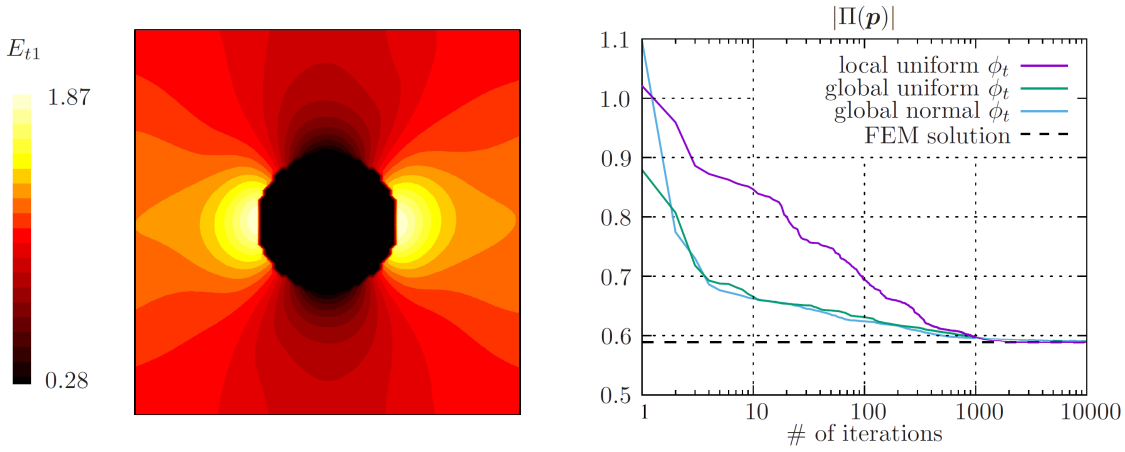

Figure 9. (**left**) Contour plot of E_{t1} for the piecewise-defined trial function in Equation (38) after 10,000 iterations in the optimization procedure; and (**right**) optimization of $\Pi(\boldsymbol{p})$ vs. iteration count for the piecewise-defined trial function with uniformly distributed weight initialization and the global trial function (33) with uniformly and normally distributed weight initialization.

4.3. Three-Dimensional Example

In the present example, we apply the method to a three-dimensional cubic RVE of unit length $l = 1$ with a spherical inclusion of radius $r_0 = 0.178l$. We first construct a global trial function, for which we need a set of seven neural networks: $N_{x_1}(x_1)$ acting in the x_1-direction, $N_{x_2}(x_2)$ acting in the x_2-direction, $N_{x_3}(x_3)$ acting in the x_3-direction, $N_{x_{12}}(x_1, x_2)$ acting in the x_1x_2-plane, $N_{x_{13}}(x_1, x_3)$

acting in the x_1x_3-plane, $N_{x_{23}}(x_2, x_3)$ acting in the x_1x_2-plane and $N_1(\boldsymbol{x})$ acting in the RVE's volume (see Figure 10). The trial function for the matrix then appears as

$$\begin{aligned}\widetilde{\phi}_t(\boldsymbol{x}, \boldsymbol{p}) = \quad & A_1\, N_1(\boldsymbol{x}, \boldsymbol{p}_1) + A_2\, N_{x_{12}}(x_1, x_2, \boldsymbol{p}_{x_{12}}) + A_3\, N_{x_{13}}(x_1, x_3, \boldsymbol{p}_{x_{13}}) \\ & + A_4\, N_{x_{23}}(x_2, x_3, \boldsymbol{p}_{x_{23}}) + A_5\, N_{x_1}(x_1, \boldsymbol{p}_{x_1}) \\ & + A_6\, N_{x_2}(x_2, \boldsymbol{p}_{x_2}) + A_7\, N_{x_3}(x_3, \boldsymbol{p}_{x_3}),\end{aligned} \tag{42}$$

where the functions A_i take the form

$$\begin{aligned} A_1 &= x_1(1-x_1)x_2(1-x_2)x_3(1-x_3), & A_5 &= x_1(1-x_1), \\ A_2 &= x_1(1-x_1)x_2(1-x_2), & A_6 &= x_2(1-x_2), \\ A_3 &= x_1(1-x_1)x_3(1-x_3), & A_7 &= x_3(1-x_3), \\ A_4 &= x_2(1-x_2)x_3(1-x_3). \end{aligned} \tag{43}$$

By construction, the trial function fulfills the periodic boundary conditions. The negative gradient of the trial function can then be computed analytically according to Equation (40) (see also Appendix D for a more detailed derivation). With the gradient at hand, we are able optimize the global potential in Equation (32) with respect to the weights of the ANNs.

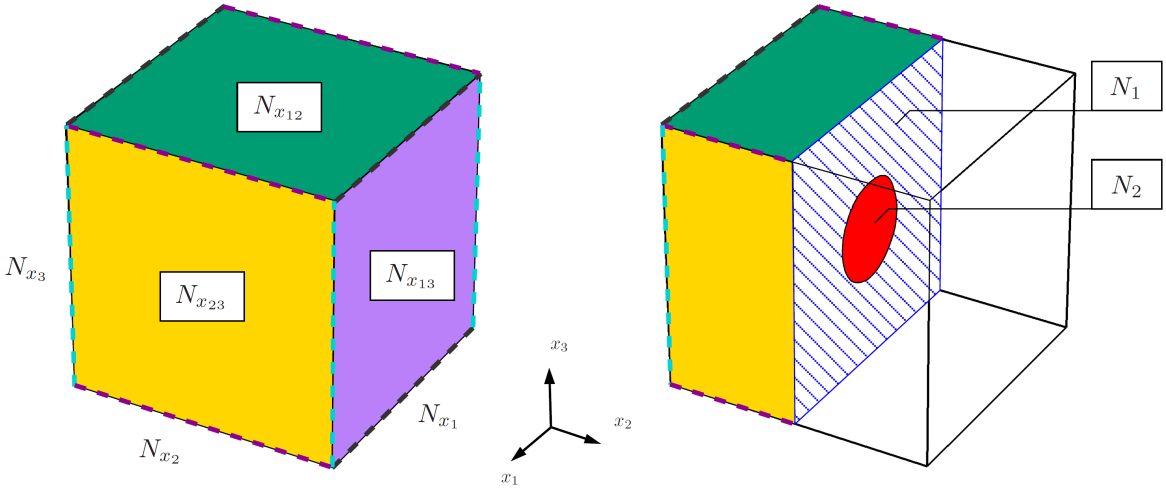

Figure 10. A representation of the artificial neural networks used in the construction of the trial function ϕ_t. There are separate ANNs for the edges, surface boundaries and volumes. In the piecewise-defined approach, there is one additional ANN acting in the inclusion.

As seen in the previous sections, the use of one global trial function might lead to oscillations in the solution field. In line with Section 4.2.2, we additionally construct a piecewise-defined trial function for comparison

$$\widetilde{\phi}_t^{\,*}(\boldsymbol{x}, \boldsymbol{p}) = \begin{cases} \begin{aligned} & A_1\, N_1(\boldsymbol{x}, \boldsymbol{p}_1) + A_2\, N_{x_{12}}(x_1, x_2, \boldsymbol{p}_{x_{12}}) + A_3\, N_{x_{13}}(x_1, x_3, \boldsymbol{p}_{x_{13}}) \\ & + A_4\, N_{x_{23}}(x_2, x_3, \boldsymbol{p}_{x_{23}}) + A_5\, N_{x_1}(x_1, \boldsymbol{p}_{x_1}) + A_6\, N_{x_2}(x_2, \boldsymbol{p}_{x_2}) \\ & + A_7\, N_{x_3}(x_2, \boldsymbol{p}_{x_3}) \end{aligned} & r \geq r_0 \\ \widetilde{\phi}_t(r_0, \varphi, \theta) + A_8\, N_2(\boldsymbol{r}, \boldsymbol{p}_2) & r < r_0. \end{cases} \tag{44}$$

where we add an eighth neural network for the inclusion and the function

$$A_8 = r_0 - r. \tag{45}$$

The transformation between the spherical coordinates $\boldsymbol{r} = (r, \varphi, \theta)$ and the Cartesian coordinates $\boldsymbol{x} = (x_1, x_2, x_3)$ is given as

$$x_1 = 0.5 + r\sin\varphi\cos\theta, \quad x_2 = 0.5 + r\sin\varphi\sin\theta, \quad x_3 = r\cos\varphi. \tag{46}$$

The gradient in Equation (40) can then be computed accordingly in Cartesian coordinates for the matrix and in spherical coordinates for the inclusion, as derived in Appendix D, allowing us to optimize the global potential

$$\overline{\Pi}^* = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|}\Pi^*(\boldsymbol{p}) = \sup_{\boldsymbol{p}} \frac{1}{|\mathcal{B}|}\left(-\int_{\mathcal{B}^{\text{matr}}} \frac{1}{2}\kappa_1 \boldsymbol{E}_t^* \cdot \boldsymbol{E}_t^* \, dV - \int_{\mathcal{B}^{\text{incl}}} \frac{1}{2}\kappa_2 \boldsymbol{E}_t^* \cdot \boldsymbol{E}_t^* \, dV\right). \tag{47}$$

For our numerical experiment, the RVE depicted in Figure 10 is loaded with a macroscopic field of $\overline{E}_1 = 1.0$, $\overline{E}_2 = 0.0$ and $\overline{E}_3 = 0.0$. The material parameters are chosen as $\kappa_1 = 1$ and $\kappa_2 = 10$. For the global approach, the mesh consists of $43 \times 43 \times 43$ equidistant integration points. The ANNs acting on the edges and surface boundaries are SLPs, having four neurons along each edge ANN and five neurons along each surface ANN. The ANN acting in the volume is a two-layer perceptron with eight neurons in each layer. This trial function totals 256 parameters to optimize in the ANNs. As for the activation function, we choose the softplus function, as it provides the best results in our examples. As for the piecewise-defined trial function in Equation (44), the ANNs are constructed in the same way. The additional ANN acting in the inclusion is an SLP with eight neurons and also has the softplus activation function. The mesh used in this case consists of 32,768 integration points in the matrix and 16,384 integration points in the inclusion. Additionally, an FFT-based simulation [23] using a conjugate gradient solver [24,39–42] is carried out for comparison, where the microstructure is discretized on a $43 \times 43 \times 43$ grid.

Figure 11 shows the contour plot of E_{t1} for the approach using the global and the piecewise-defined trial function after 20,000 iterations. In comparison to the FFT-based solution, both simulations produce qualitatively good results. The global approximation scheme captures the expected jump of E_{t1} across the phase interface a little less distinctly compared with the piecewise-defined one. However, the piecewise-defined approximation is more difficult to construct and will be quite challenging to implement in the case of complicated microstructure geometries. Quantitatively, the optimized global potentials after 20,000 iterations are quite close to each other, with $|\overline{\Pi}|(\boldsymbol{p}) = 0.529926$ for the global approach and $|\overline{\Pi}|^*(\boldsymbol{p}) = 0.529692$ for the piecewise-defined approach, compared to a optimum energy $|\overline{\Pi}|^{\text{FFT}} = 0.527590$ obtained by the FFT-based simulation.

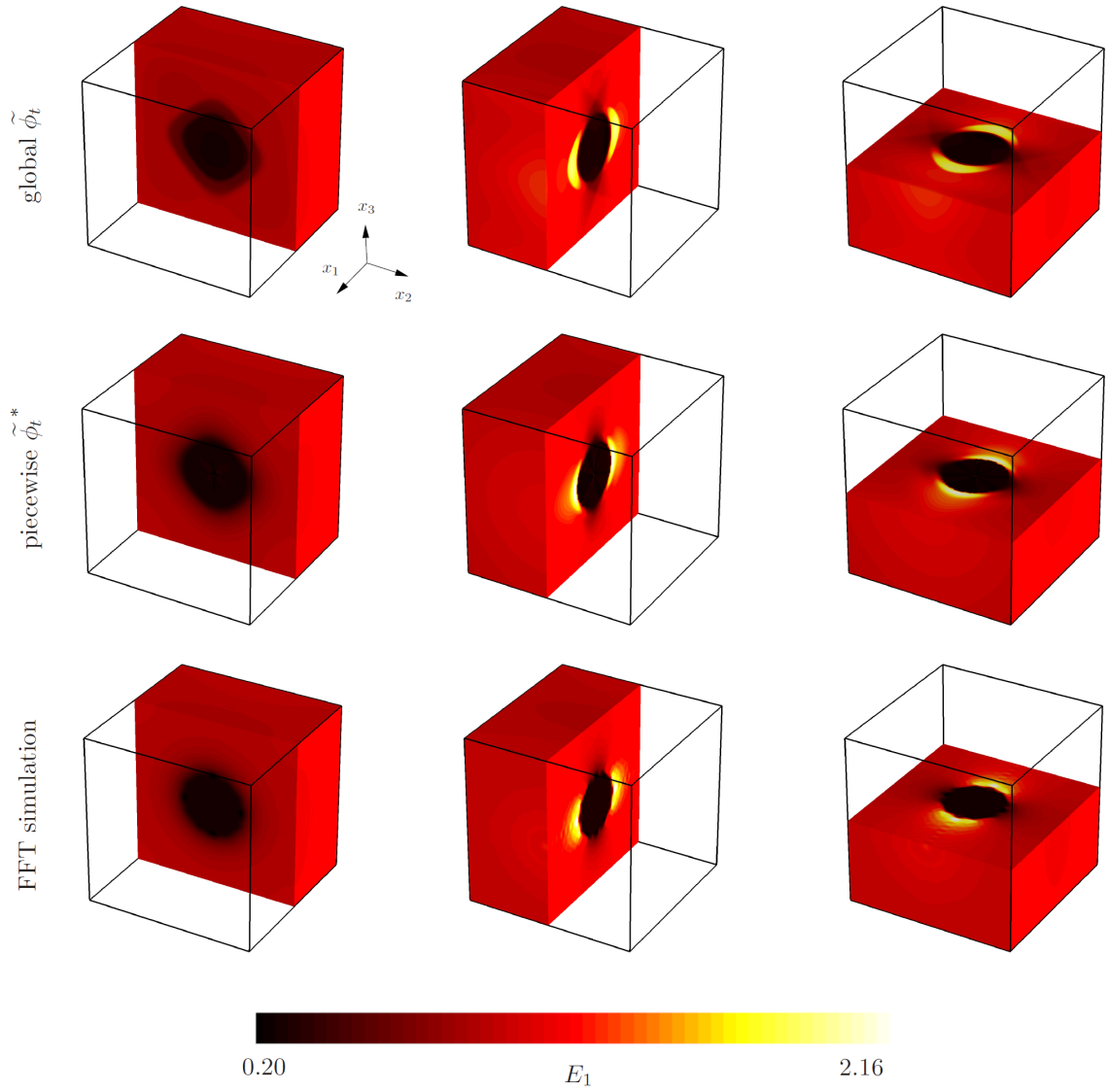

Figure 11. Contour plot of E_1 for a global and a piecewise-defined construction of the trial function as well as an FFT-based simulation for comparison. The results of the ANN simulations are close to the FFT-based simulation. The jump discontinuity is more distinct in the piecewise-defined approach compared with the global approach.

5. Conclusions

We presented a solution scheme to periodic boundary value problems in homogenization based on the training of artificial neural networks. They were employed in tandem with the multiplicative factors in a way that the resulted trial function fulfilled the boundary conditions a priori and thus we arrived at the unconstrained optimization problem. The numerical examples showed that physically reasonable results can be obtained with rather low amounts of neurons, which allows for low memory demand. A construction of trial functions by defining them piecewise in separate sub-domains led to lower oscillations and a generally more stable training behavior compared with a global approach but was geometrically more challenging to construct. The scheme carried over to three dimensions quite well. We assume this to stem from the ratio of neurons compared with the number of integration points: In the considered example of a cube-shaped matrix with spherical inclusion, the solution could be approximated using a quite low neuron count while the number of integration points grew a lot. For future progress, the training speed of the neural network needs to be improved. More sophisticated

ANN structures such as deep nets or recurrent nets might further improve the approximation and training behavior of the method, while methods such as dropout or regularization might assist to avoid problems of overfitting.

Author Contributions: Conceptualization, F.S.G., L.T.K.N. and M.-A.K.; Funding acquisition, M.-A.K.; Investigation, F.S.G.; Methodology, F.S.G., L.T.K.N. and M.-A.K.; Software, F.S.G. and L.T.K.N.; Validation, F.S.G., L.T.K.N. and M.-A.K.; Visualization, F.S.G.; Writing—original draft, F.S.G.; Writing—review & editing, L.T.K.N. and M.-A.K.

Acknowledgments: Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC 2075—390740016.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Matlab Script for Global One-Dimensional Example

```
% Evaluation points in area
dx = 0.01;
x = dx/2:dx:1-dx/2;

% Applied macroscopic electric field
E0 = 0.01;

% Electric permittivity in laminate
KK = zeros(size(x));
KK(x > 1/2) = 2.0;
KK(x < 1/2) = 1.0;

% Number of hidden neurons
nn = 10;

% Initialize weights and biases
wb0 = rand(3*nn+1,1);

% Anonymous function handle
f = @(wb)neuralapprox(wb,KK,E0,x,dx,nn);

% Call unconstrained minimizer
opts = optimoptions(@fminunc,'Algorithm','quasi-newton',...
    'StepTolerance',1e-12,'OptimalityTolerance',1e-12,...
    'MaxFunctionEvaluations',10000,...
    'SpecifyObjectiveGradient',true,'CheckGradients',true,...
    'FiniteDifferenceType','central','MaxIterations',10000);
[wb,f] = fminunc(f,wb0,opts)

function [f, g] = neuralapprox(wb,KK,E0,x,dx,nn)

% restore weights and biases for ANN
u = wb(1:nn);
w = wb(nn+1:2*nn);
b = wb(2*nn+1);
v = wb(2*nn+2:3*nn+1);

% Neural network response for feedforward net (FFN) (Eq.(13))
```

```
z = w*x + u;
sigmoid = (1 + exp(-z)).^(-1);
ANN = sum(v.*sigmoid) + b;

% Spatial derivative response for feedforward net (Eq.(14))
d_sigmoid = sigmoid.*(1 - sigmoid);
d_ANN = sum(v.*w.*d_sigmoid);

% Derivative of 1st FFN and dFFN/dx w.r.t. its weights (Eq.(15))
d2_sigmoid = sigmoid.*(1 - sigmoid).^2 - sigmoid.^2.*(1 - sigmoid);

xmatr = x.*ones(size(z));
ANN_du = v.*d_sigmoid;
ANN_dw = v.*d_sigmoid.*xmatr;
ANN_db = ones(size(x));
ANN_dv = sigmoid;

d_ANN_du = v.*w.*d2_sigmoid;
d_ANN_dw = v.*d_sigmoid + v.*w.*d2_sigmoid.*xmatr;
d_ANN_db = zeros(size(x));
d_ANN_dv = w.*d_sigmoid;

% Derivatives of the trial function phi = x*(1-x)*N (Eq.(24)&(15))
d_phi_t  = (1-2*x).*ANN + x.*(1-x).*d_ANN;
d_phi_t_du  = (1-2*xmatr).*ANN_du + xmatr.*(1-xmatr).*d_ANN_du;
d_phi_t_dw  = (1-2*xmatr).*ANN_dw + xmatr.*(1-xmatr).*d_ANN_dw;
d_phi_t_db  = (1-2*x).*ANN_db + x.*(1-x).*d_ANN_db;
d_phi_t_dv  = (1-2*xmatr).*ANN_dv + xmatr.*(1-xmatr).*d_ANN_dv;

% Cost function and its derivatives w.r.t. the weights (Eq.(26))
JJ = 0.5*KK.*(E0 - d_phi_t).^2;

JJ_du = -KK.*(E0 - d_phi_t).*d_phi_t_du;
JJ_dw = -KK.*(E0 - d_phi_t).*d_phi_t_dw;
JJ_db = -KK.*(E0 - d_phi_t).*d_phi_t_db;
JJ_dv = -KK.*(E0 - d_phi_t).*d_phi_t_dv;

g1 = [JJ_du;JJ_dw;JJ_db;JJ_dv];

f = dx*sum(JJ);
g = dx*sum(g1, 2);

end
```

Appendix B. Matlab Script for Piecewise-Defined One-Dimensional Example

```
% Evaluation points in area
dx = 0.01;
x = dx/2:dx:1-dx/2;

% Applied macroscopic load
E0 = 0.01;

% Electric permittivity in laminate
KK = zeros(size(x));
KK(x > 1/2) = 2.0;
KK(x < 1/2) = 1.0;

% Number of hidden neurons
nn = 10;

% Initialize weights and biases
wb0 = rand(6*nn+2,1)*(E0/nn);

% Anonymous function handle
f = @(wb)neuralapprox(wb,KK,E0,x,dx,nn);

% Call unconstrained minimizer
opts = optimoptions(@fminunc,'Algorithm','quasi-newton',...
    'SpecifyObjectiveGradient',true,'CheckGradients',true,...
    'FiniteDifferenceType','central','MaxIterations',5000);
[wb,f] = fminunc(f,wb0,opts)

function [f, g] = neuralapprox(wb,KK,E0,xx,dx,nn)
%%% Cost function for interval [0,0.5]

% Restore weights and biases for first ANN
u = wb(1:nn);
w = wb(nn+1:2*nn);
b = wb(2*nn+1);
v = wb(2*nn+2:3*nn+1);

% Quadrature points for integration
x = xx(1:floor(size(xx,2)/2));
K = KK(1:floor(size(KK,2)/2));

% Neural network response for 1st feedforward net (FFN) (Eq.(13))
z = w*x + u;
sigmoid = (1 + exp(-z)).^(-1);
ANN = sum(v.*sigmoid) + b;

% Spatial derivative response for 1st feedforward net (Eq.(14))
d_sigmoid = sigmoid.*(1 - sigmoid);
d_ANN = sum(v.*w.*d_sigmoid);
```

```

% Derivative of 1st FFN and dFFN/dx w.r.t. its weights (Eq.(15))
d2_sigmoid = sigmoid.*(1 - sigmoid).^2 - sigmoid.^2.*(1 - sigmoid);

xmatr = x.*ones(size(z));

ANN_du = v.*d_sigmoid;
ANN_dw = v.*d_sigmoid.*xmatr;
ANN_db = ones(size(x));
ANN_dv = sigmoid;

d_ANN_du = v.*w.*d2_sigmoid;
d_ANN_dw = v.*d_sigmoid + v.*w.*d2_sigmoid.*xmatr;
d_ANN_db = zeros(size(x));
d_ANN_dv = w.*d_sigmoid;

% FFN and derivatives evaluated at the discontinuity (Eq.(28)&(15))
z_disc = w*0.5 + u;

sigmoid_disc = (1 + exp(-z_disc)).^(-1);
d_sigmoid_disc = sigmoid_disc.*(1 - sigmoid_disc);

ANN_disc = sum(v.*sigmoid_disc) + b;
ANN_disc_du = v.*d_sigmoid_disc;
ANN_disc_dw = v.*d_sigmoid_disc*0.5;
ANN_disc_db = 1.0;
ANN_disc_dv = sigmoid_disc;

% Derivatives of the trial function phi = x*N_1 (Eq.(28)&(15))
phi_t_disc    = 0.5.*ANN_disc;
d_phi_t_disc_du = ANN_disc_du;
d_phi_t_disc_dw = ANN_disc_dw;
d_phi_t_disc_db = ANN_disc_db;
d_phi_t_disc_dv = ANN_disc_dv;

d_phi_t  = ANN + x.*d_ANN;

d_phi_t_du  = ANN_du + xmatr.*d_ANN_du;
d_phi_t_dw  = ANN_dw + xmatr.*d_ANN_dw;
d_phi_t_db  = ANN_db + x.*d_ANN_db;
d_phi_t_dv  = ANN_dv + xmatr.*d_ANN_dv;

% Cost function and its derivatives w.r.t. the weights (Eq.(30))
JJ = 0.5*K.*(E0 - d_phi_t).^2;

JJ_du = -K.*(E0 - d_phi_t).*d_phi_t_du;
JJ_dw = -K.*(E0 - d_phi_t).*d_phi_t_dw;
JJ_db = -K.*(E0 - d_phi_t).*d_phi_t_db;
JJ_dv = -K.*(E0 - d_phi_t).*d_phi_t_dv;

```

```
g1 = [JJ_du;JJ_dw;JJ_db;JJ_dv];

f = dx*sum(JJ);
g = dx*sum(g1, 2);

%%% Cost function for interval [0.5,1.0]

% Restore weights and biases for second ANN
u = wb(3*nn+2:4*nn+1);
w = wb(4*nn+2:5*nn+1);
b = wb(5*nn+2);
v = wb(5*nn+3:6*nn+2);

% Quadrature points for integration
x = xx(floor(size(xx,2)/2)+1:end);
K = KK(floor(size(KK,2)/2)+1:end);

% Neural network response for 2nd feedforward net (FFN) (Eq.(13))
z = w*x + u;
sigmoid = (1 + exp(-z)).^(-1);
ANN = sum(v.*sigmoid) + b;

% Spatial derivative response for 2nd feedforward net (Eq.(14))
d_sigmoid = sigmoid.*(1 - sigmoid);
d_ANN = sum(v.*w.*d_sigmoid);

% Derivative of end FFN and dFFN/dx w.r.t. its weights (Eq.(28)&(15))
d2_sigmoid = sigmoid.*(1 - sigmoid).^2 - sigmoid.^2.*(1 - sigmoid);

xmatr = x.*ones(size(z));

ANN_du = v.*d_sigmoid;
ANN_dw = v.*d_sigmoid.*xmatr;
ANN_db = ones(size(x));
ANN_dv = sigmoid;

d_ANN_du = v.*w.*d2_sigmoid;
d_ANN_dw = v.*d_sigmoid + v.*w.*d2_sigmoid.*xmatr;
d_ANN_db = zeros(size(x));
d_ANN_dv = w.*d_sigmoid;

% Derivatives of the trial function
% phi = (0.5-x)*(1-x)*N_2 + 2*(1-x)*phi(0.5) (Eq.(28)&(15))
d_phi_t  = (2*x-1.5).*ANN + (0.5-x).*(1-x).*d_ANN - 2*phi_t_disc;

d_phi_t_du  = (2*xmatr-1.5).*ANN_du + (0.5-xmatr).*(1-xmatr).*d_ANN_du;
d_phi_t_dw  = (2*xmatr-1.5).*ANN_dw + (0.5-xmatr).*(1-xmatr).*d_ANN_dw;
d_phi_t_db  = (2*x-1.5).*ANN_db     + (0.5-x).*(1-x).*d_ANN_db;
d_phi_t_dv  = (2*xmatr-1.5).*ANN_dv + (0.5-xmatr).*(1-xmatr).*d_ANN_dv;
```

```
% Cost function and its derivatives w.r.t. the weights (Eq.(30))
JJ = 0.5*K.*(E0 - d_phi_t).^2;

JJ_du = -K.*(E0 - d_phi_t).*d_phi_t_du;
JJ_dw = -K.*(E0 - d_phi_t).*d_phi_t_dw;
JJ_db = -K.*(E0 - d_phi_t).*d_phi_t_db;
JJ_dv = -K.*(E0 - d_phi_t).*d_phi_t_dv;

JJ_disc_du = -K.*(E0 - d_phi_t).*d_phi_t_disc_du;
JJ_disc_dw = -K.*(E0 - d_phi_t).*d_phi_t_disc_dw;
JJ_disc_db = -K.*(E0 - d_phi_t).*d_phi_t_disc_db;
JJ_disc_dv = -K.*(E0 - d_phi_t).*d_phi_t_disc_dv;

g0 = [JJ_disc_du;JJ_disc_dw;JJ_disc_db;JJ_disc_dv];
g1 = [JJ_du;JJ_dw;JJ_db;JJ_dv];

f = f + dx*sum(JJ);
g = g - dx*sum(g0, 2);
g = [g;dx*sum(g1, 2)];
end
```

Appendix C. Two-Dimensional Trial Function and Derivatives

Recalling Section 4.2, we have the trial function in the matrix and the inclusion defined as

$$\widetilde{\phi}_t = \begin{cases} x_1(1-x_1)x_2(1-x_2)N_1(\boldsymbol{x},\boldsymbol{p}_1) + x_1(1-x_1)N_{x_1}(x_1,\boldsymbol{p}_{x_1}) & r \geq r_0 \\ +x_2(1-x_2)N_{x_2}(x_2,\boldsymbol{p}_{x_2}) & \\ \widetilde{\phi}_t(r_0,\varphi) + (r_0 - r)N_2(\boldsymbol{r},\boldsymbol{p}_1) & r < r_0 \end{cases}, \tag{A1}$$

where the coordinate transformation between Cartesian and polar coordinates is performed as

$$x_1 = 0.5 + r\cos\varphi \quad \text{and} \quad x_2 = 0.5 + r\sin\varphi. \tag{A2}$$

The negative gradient of the trial function in the matrix ($r \geq r_0$) in Cartesian coordinates then appears as

$$\begin{aligned} \widetilde{\boldsymbol{E}}_t = -\nabla\widetilde{\phi}_t = &- \begin{bmatrix} (1-2x_1)x_2(1-x_2)N_1 + x_1(1-x_1)x_2(1-x_2)\frac{\partial N_1}{\partial x_1} \\ (1-2x_2)x_1(1-x_1)N_1 + x_1(1-x_1)x_2(1-x_2)\frac{\partial N_1}{\partial x_2} \end{bmatrix} \\ &- \begin{bmatrix} (1-2x_1)N_{x_1} + x_1(1-x_1)\frac{\partial N_{x_1}}{\partial x_1} \\ (1-2x_2)N_{x_2} + x_2(1-x_2)\frac{\partial N_{x_2}}{\partial x_2} \end{bmatrix}. \end{aligned} \tag{A3}$$

The gradient of the trial function in the inclusion ($r < r_0$) in polar coordinates can be computed through

$$\widetilde{\boldsymbol{E}}_t = -\nabla_{\boldsymbol{r}}\widetilde{\phi}_t = -\boldsymbol{J}_0^T \cdot \begin{bmatrix} \frac{\partial\widetilde{\phi}_t}{\partial x_1} \\ \frac{\partial\widetilde{\phi}_t}{\partial x_2} \end{bmatrix}_{r_0,\varphi} - \begin{bmatrix} -N_2 + (r_0 - r)\frac{\partial N_2}{\partial r} \\ \frac{r_0 - r}{r}\frac{\partial N_2}{\partial\varphi} \end{bmatrix}, \tag{A4}$$

where the derivatives with respect to x_1 and x_2 are evaluated at $x_1(r_0, \varphi)$ and $x_2(r_0, \varphi)$. The Jacobian for a radius of r_0 takes the form

$$\boldsymbol{J}_0 = \begin{bmatrix} 0 & -\frac{r_0}{r}\sin\varphi \\ 0 & \frac{r_0}{r}\cos\varphi \end{bmatrix}. \tag{A5}$$

The integration of the global potential in Equation (41) is then piecewise carried out

$$\Pi(\boldsymbol{p}) = \int_{\mathcal{B}^{\text{matr}}} \frac{1}{2}\kappa_1 \boldsymbol{E}_t \cdot \boldsymbol{E}_t \, dx \, dy + \int_{\mathcal{B}^{\text{incl}}} \frac{1}{2}\kappa_2 \boldsymbol{E}_t \cdot \boldsymbol{E}_t \, r \, dr \, d\varphi, \tag{A6}$$

where the integration for the matrix is carried out in Cartesian coordinates and the integration for the inclusion is carried out in polar coordinates.

Appendix D. Three-Dimensional Trial Function and Derivatives

Recalling Section 4.3, the piecewise-defined trial function is constructed as

$$\widetilde{\phi_t}^* = \begin{cases} A_1 N_1(\boldsymbol{x}, \boldsymbol{p}_1) + A_2 N_{x_{12}}(x_1, x_2, \boldsymbol{p}_{x_{12}}) + A_3 N_{x_{13}}(x_1, x_3, \boldsymbol{p}_{x_{13}}) & \\ +A_4 N_{x_{23}}(x_2, x_3, \boldsymbol{p}_{x_{23}}) + A_5 N_{x_1}(x_1, \boldsymbol{p}_{x_1}) + A_6 N_{x_2}(x_2, \boldsymbol{p}_{x_2}) & r \geq r_0 \\ +A_7 N_{x_3}(x_3, \boldsymbol{p}_{x_3}) & \\ \widetilde{\phi_t}(r_0, \varphi, \theta) + A_8 N_2(\boldsymbol{r}, \boldsymbol{p}_2) & r < r_0 \end{cases}, \tag{A7}$$

where the factors

$$\begin{aligned} A_1 &= x_1(1-x_1)x_2(1-x_2)x_3(1-x_3), & A_5 &= x_1(1-x_1), \\ A_2 &= x_1(1-x_1)x_2(1-x_2), & A_6 &= x_2(1-x_2), \\ A_3 &= x_1(1-x_1)x_3(1-x_3), & A_7 &= x_3(1-x_3), \\ A_4 &= x_2(1-x_2)x_3(1-x_3). & A_8 &= r - r_0 \end{aligned} \tag{A8}$$

ensure the satisfaction of the boundary conditions. As the matrix material ($r \geq r_0$) is described in Cartesian coordinates, the gradient can be straightforward computed as

$$\begin{aligned} \widetilde{\boldsymbol{E}}_t^* = -\nabla\widetilde{\phi}_t = &- \begin{bmatrix} \frac{\partial A_1}{\partial x_1} N_1 + A_1 \frac{\partial N_1}{\partial x_1} \\ \frac{\partial A_1}{\partial x_2} N_1 + A_1 \frac{\partial N_1}{\partial x_2} \\ \frac{\partial A_1}{\partial x_3} N_1 + A_1 \frac{\partial N_1}{\partial x_3} \end{bmatrix} - \begin{bmatrix} \frac{\partial A_2}{\partial x_1} N_{x_{12}} + A_2 \frac{\partial N_{x_{12}}}{\partial x_1} \\ \frac{\partial A_2}{\partial x_2} N_{x_{12}} + A_2 \frac{\partial N_{x_{12}}}{\partial x_2} \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{\partial A_3}{\partial x_1} N_{x_{13}} + A_3 \frac{\partial N_{x_{13}}}{\partial x_1} \\ 0 \\ \frac{\partial A_3}{\partial x_3} N_{x_{13}} + A_3 \frac{\partial N_{x_{13}}}{\partial x_3} \end{bmatrix} \\ &- \begin{bmatrix} 0 \\ \frac{\partial A_4}{\partial x_2} N_{x_{23}} + A_4 \frac{\partial N_{x_{23}}}{\partial x_2} \\ \frac{\partial A_4}{\partial x_3} N_{x_{23}} + A_4 \frac{\partial N_{x_{23}}}{\partial x_3} \end{bmatrix} - \begin{bmatrix} \frac{\partial A_5}{\partial x_1} N_{x_1} + A_5 \frac{\partial N_{x_1}}{\partial x_1} \\ \frac{\partial A_6}{\partial x_2} N_{x_2} + A_6 \frac{\partial N_{x_2}}{\partial x_2} \\ \frac{\partial A_7}{\partial x_3} N_{x_3} + A_7 \frac{\partial N_{x_3}}{\partial x_3} \end{bmatrix}, \end{aligned} \tag{A9}$$

where the derivatives of the factors A_i take the form

$$\begin{aligned}
\frac{\partial A_1}{\partial x_1} &= (1-2x_1)x_2(1-x_2)x_3(1-x_3), & \frac{\partial A_1}{\partial x_2} &= (1-2x_2)x_1(1-x_1)x_3(1-x_3),\\
\frac{\partial A_1}{\partial x_3} &= (1-2x_3)x_1(1-x_1)x_2(1-x_2), & \frac{\partial A_2}{\partial x_1} &= (1-2x_1)x_2(1-x_2),\\
\frac{\partial A_2}{\partial x_2} &= (1-2x_2)x_1(1-x_1), & \frac{\partial A_3}{\partial x_1} &= (1-2x_1)x_3(1-x_3),\\
\frac{\partial A_3}{\partial x_3} &= (1-2x_3)x_1(1-x_1), & \frac{\partial A_4}{\partial x_2} &= (1-2x_2)x_3(1-x_3),\\
\frac{\partial A_4}{\partial x_3} &= (1-2x_3)x_2(1-x_2), & \frac{\partial A_5}{\partial x_1} &= (1-2x_1),\\
\frac{\partial A_6}{\partial x_2} &= (1-2x_2), & \frac{\partial A_7}{\partial x_3} &= (1-2x_3).
\end{aligned} \tag{A10}$$

As for the inclusion, the transformation between the spherical coordinates $\boldsymbol{r} = (r, \varphi, \theta)$ and the Cartesian coordinates $\boldsymbol{x} = (x_1, x_2, x_3)$ is given as

$$x_1 = 0.5 + r\sin\varphi\cos\theta, \quad x_2 = 0.5 + r\sin\varphi\sin\theta \quad \text{and} \quad x_3 = r\cos\varphi. \tag{A11}$$

One could now either directly substitute the latter transformation into the trial function in Equation (A7) and take the derivatives with respect to $\boldsymbol{r}$. Alternatively, the Jacobian for the matrix can be computed as

$$\boldsymbol{J}_0 = \begin{bmatrix} 0 & \frac{r_0}{r}\cos\varphi\cos\theta & -\frac{r_0}{r}\sin\theta \\ 0 & \frac{r_0}{r}\cos\varphi\sin\theta & \frac{r_0}{r}\cos\theta \\ 0 & -\frac{r_0}{r}\sin\varphi & 0 \end{bmatrix}, \tag{A12}$$

which allows for the computation of the gradient of ϕ_t^* in spherical coordinates through

$$\widetilde{\boldsymbol{E}}_t = -\nabla_{\boldsymbol{r}}\widetilde{\phi}_t = -\boldsymbol{J}_0^T \cdot \begin{bmatrix} \frac{\partial \widetilde{\phi}_t}{\partial x_1} \\ \frac{\partial \widetilde{\phi}_t}{\partial x_2} \\ \frac{\partial \widetilde{\phi}_t}{\partial x_3} \end{bmatrix}_{r_0,\varphi,\theta} - \begin{bmatrix} \frac{\partial A_8}{\partial r} N_2 + A_8 \frac{\partial N_2}{\partial r} \\ \frac{A_8}{r}\frac{\partial N_2}{\partial \varphi} \\ \frac{A_8}{r\sin\varphi}\frac{\partial N_2}{\partial \theta} \end{bmatrix}. \tag{A13}$$

Here, the partial derivative is simply $\partial A_8/\partial r = -1$. The integration of the global potential in Equation (41) is then piecewise carried out

$$\Pi(\boldsymbol{p}) = \int_{\mathcal{B}^{\text{matr}}} \frac{1}{2}\kappa_1 \boldsymbol{E}_t \cdot \boldsymbol{E}_t \, dx_1 \, dx_2 \, dx_3 + \int_{\mathcal{B}^{\text{incl}}} \frac{1}{2}\kappa_2 \boldsymbol{E}_t \cdot \boldsymbol{E}_t \, r^2 \sin\varphi \, dr \, d\varphi \, d\theta, \tag{A14}$$

where one has to appropriately add the constant macroscopic loads to the fluctuations in Equations (A9) and (A13) in Cartesian and polar coordinates.

References

1. Hebb, D.O. *The Organization of Behavior*; John Wiley & Sons: New York, NY, USA, 1949.
2. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
3. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: New York, NY, USA, 2012; pp. 1097–1105.

4. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
5. Denning, D.E. An intrusion-detection model. *IEEE Trans. Softw. Eng.* **1987**, *2*, 222–232. [CrossRef]
6. Steinwart, I.; Christmann, A. *Support Vector Machines*; Springer Science & Business Media: Cham, Switzerland, 2008.
7. Raissi, M.; Yazdani, A.; Karniadakis, G.E. Hidden fluid mechanics: A Navier–Stokes informed deep learning framework for assimilating flow visualization data. Available online: http://arxiv.org/abs/1808.04327 (accessed on 12 April 2019).
8. Ghaboussi, J.; Garrett, J.H.; Wu, X. Knowledge-based modeling of material behavior with neural networks. *J. Eng. Mech.* **1991**, *117*, 132–153. [CrossRef]
9. Huber, N.; Tsakmakis, C. A neural network tool for identifying the material parameters of a finite deformation viscoplasticity model with static recovery. *Comput. Methods Appl. Mech. Eng.* **2001**, *191*, 353–384. [CrossRef]
10. Le, B.; Yvonnet, J.; He, Q.-C. Computational homogenization of nonlinear elastic materials using neural networks. *Int. J. Numer. Methods Eng.* **2015**, *104*, 1061–1084. [CrossRef]
11. Bessa, M.; Bostanabad, R.; Liu, Z.; Hu, A.; Apley, D.W.; Brinson, C.; Chen, W.; Liu, W. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Comput. Methods Appl. Mech. Eng.* **2017**, *320*, 633–667. [CrossRef]
12. Liu, Z.; Bessa, M.; Liu, W.K. Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials. *Comput. Methods Appl. Mech. Eng.* **2016**, *306*, 319–341. [CrossRef]
13. Voigt, W. *Theoretische Studien über die Elastizitastsverhältnisse der Cristalle*; Königliche Gesellschaft der Wissenschaften zu Göttingen: Göttingen, Germany, 1887.
14. Reuss, A. Berechnung der fließgrenze von mischkristallen auf grund der plastizitätsbedingung für einkristalle. *Zeitschrift für Angewandte Mathematik und Mechanik* **1929**, *9*, 49–58. [CrossRef]
15. Hill, R. The elastic behaviour of a crystalline aggregate. *Proc. Phys. Soc.* **1952**, *65*, 349. [CrossRef]
16. Hashin, Z.; Shtrikman, S. A variational approach to the theory of the elastic behaviour of multiphase materials. *J. Mech. Phys. Solids* **1963**, *11*, 127–140. [CrossRef]
17. Willis, J. Bounds and self-consistent estimates for the overall properties of anisotropic composites. *J. Mech. Phys. Solids* **1977**, *25*, 185–202. [CrossRef]
18. Budiansky, B. On the elastic moduli of some heterogeneous materials. *J. Mech. Phys. Solids* **1965**, *13*, 223–227. [CrossRef]
19. Hill, R. A self-consistent mechanics of composite materials. *J. Mech. Phys. Solids* **1965**, *13*, 213–222. [CrossRef]
20. Mori, T.; Tanaka, K. Average stress in matrix and average elastic energy of materials with misfitting inclusions. *Acta Mech.* **1973**, *21*, 571–574. [CrossRef]
21. Miehe, C.; Schröder, J.; Schotte, J. Computational homogenization analysis in finite plasticity. Simulation of texture development in polycrystalline materials. *Comput. Methods Appl. Mech. Eng.* **1999**, *171*, 387–418. [CrossRef]
22. Schröder, J. A numerical two-scale homogenization scheme: The FE^2–method. In *Plasticity and Beyond*; Schröder, J., Hackl, K., Eds.; Springer: Cham, Switzerland, 2014; pp. 1–64.
23. Moulinec, H.; Suquet, P. A fast numerical method for computing the linear and nonlinear mechanical properties of composites. *Académie des Sciences* **1994**, *2*, 1417–1423.
24. Zeman, J.; Vondřejc, J.; Novak, J.; Marek, I. Accelerating a fft-based solver for numerical homogenization of periodic media by conjugate gradients. *J. Comput. Phys.* **2010**, *229*, 8065–8071. [CrossRef]
25. Fritzen, F.; Leuschner, M. Reduced basis hybrid computational homogenization based on a mixed incremental formulation. *Comput. Methods Appl. Mech. Eng.* **2013**, *260*, 143–154. [CrossRef]
26. Ryckelynck, D. A priori hyperreduction method: An adaptive approach. *J. Comput. Phys.* **2005**, *202*, 346–366. [CrossRef]
27. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
28. Hill, R. Elastic properties of reinforced solids: Some theoretical principles. *J. Mech. Phys. Solids* **1963**, *11*, 357–372. [CrossRef]
29. Jackson, J.D. *Classical Electrodynamics*; Wiley: Hoboken, NJ, USA, 1999.
30. Bensoussan, A.; Lions, J.-L.; Papanicolaou, G. *Asymptotic Analysis for Periodic Structures*; American Mathematical Society: Providence, RI, USA, 2011; Volume 374.

31. Nemat-Nasser, S.; Lori, M.; Datta, S. Micromechanics: Overall properties of heterogeneous materials. *J. Appl. Mech.* **1996**, *63*, 561. [CrossRef]
32. Terada, K.; Hori, M.; Kyoya, T.; Kikuchi, N. Simulation of the multi-scale convergence in computational homogenization approaches. *Int. J. Solids Struct.* **2000**, *37*, 2285–2311. [CrossRef]
33. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
34. MATLAB R2017b (Version 9.3.0). The MathWorks Inc.: Natick, MA, USA, 2017. Available online: https://www.mathworks.com/products/matlab.html (accessed on 12 April 2019).
35. Rumelhart, D.E.; McClelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*; MIT Press: Cambridge, MA, USA, 1986.
36. Byrd, R.H.; Lu, P.; Nocedal, J.; Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* **1995**, *16*, 1190–1208. [CrossRef]
37. Zhu, C.; Byrd, R.H.; Lu, P.; Nocedal, J. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* **1997**, *23*, 550–560. [CrossRef]
38. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
39. Gelebart, L.; Mondon-Cancel, R. Non-linear extension of fft-based methods accelerated by conjugate gradients to evaluate the mechanical behavior of composite materials. *Comput. Mat. Sci.* **2013**, *77*, 430–439. [CrossRef]
40. Göküzüm, F.S.; Keip, M.-A. An algorithmically consistent macroscopic tangent operator for FFT-based computational homogenization. *Int. J. Numer. Methods Eng.* **2018**, *113*, 581–600. [CrossRef]
41. Göküzüm, F.S.; Nguyen, L.T.K.; Keip, M.A. A multiscale fe-fft framework for electro-active materials at finite strains. *Comput. Mech.* **2019**, 1–22. [CrossRef]
42. Kabel, M.; Böhlke, T.; Schneider, M. Efficient fixed point and newton-krylov solvers for fft-based homogenization of elasticity at large deformations. *Comput. Mech.* **2014**, *54*, 1497–1514. [CrossRef]