J. Sens. Sens. Syst., 6, 269–284, 2017
https://doi.org/10.5194/jsss-6-269-2017

JSSS Journal of Sensors and Sensor Systems Open Access

Accelerated optimizations of an electromagnetic acoustic transducer with artificial neural networks as metamodels

Shen Wang[1], Songling Huang[1], Qing Wang[2], Lisha Peng[1], and Wei Zhao[1]

[1]State Key Lab. of Power System, Dept. of Electrical Engineering, Tsinghua University, Beijing 100084, China
[2]School of Engineering and Computing Sciences, Durham University, DH1 3LE, Durham, UK

Correspondence to: Shen Wang (wangshen@mail.tsinghua.edu.cn)

Received: 21 March 2017 – Revised: 7 July 2017 – Accepted: 14 July 2017 – Published: 16 August 2017

Abstract. Electromagnetic acoustic transducers (EMATs) are noncontact transducers generating ultrasonic waves directly in the conductive sample. Despite the advantages, their transduction efficiencies are relatively low, so it is imperative to build accurate multiphysics models of EMATs and optimize the structural parameters accordingly, using a suitable optimization algorithm. The optimizing process often involves a large number of runs of the computationally expensive numerical models, so metamodels as substitutes for the real numerical models are helpful for the optimizations. In this work the focus is on the artificial neural networks as the metamodels of an omnidirectional EMAT, including the multilayer feedforward networks trained with the basic and improved back propagation algorithms and the radial basis function networks with exact and nonexact interpolations. The developed neural-network programs are tested on an example problem. Then the model of an omnidirectional EMAT generating Lamb waves in a linearized steel plate is introduced, and various approaches to calculate the amplitudes of the displacement component waveforms are discussed. The neural-network metamodels are then built for the EMAT model and compared to the displacement component amplitude (or ratio of amplitudes) surface data on a discrete grid of the design variables as the reference, applying a multifrequency model with FFT (fast Fourier transform)/IFFT (inverse FFT) processing. Finally the two-objective optimization problem is formulated with one objective function minimizing the ratio of the amplitude of the S0-mode Lamb wave to that of the A0 mode, and the other objective function minimizing as the negative amplitude of the A0 mode. Pareto fronts in the criterion space are solved with the neural-network models and the total time consumption is greatly decreased. From the study it could be observed that the radial basis function network with exact interpolation has the best performance considering its accuracy of approximation and the time required to build the metamodel.

1 Introduction

Nondestructive testing & evaluation (NDT & E) is the field in which different methods based on quite different physical mechanisms are employed to check critical components in various structures for defects or other potential threats, without harming the components or the structures in any way. The testing methods diversely include electromagnetic methods, ultrasonic methods, X-rays and so on. Structural health monitoring (SHM) is a closely related field where NDT & E methods could be applied to collect information about parameters related to structural performance. Since large-scale structures occupy large areas, in SHM the monitoring system is often distributed in nature, and a number of sensors or transducers are used (Gao et al., 2016). In both NDT & E systems and SHM systems where ultrasonic waves are employed, the ultrasonic sensors or transducers are basic and at the same time critical components. Traditionally the piezoelectric transducers are used to generate and receive ultrasonic waves, while they rely on liquid coupling to transfer the mechanical energy into the components under investigation, and sometimes this requirement of coupling is not convenient. So some noncontact transduction methods have become popular in recent years for NDT & E and SHM applications based on ultrasonic waves.

Electromagnetic acoustic transducers (EMATs) are new transducers relying on electromagnetic effects to generate ultrasonic waves directly in the tested metal samples, according to Lorentz force or magnetostriction mechanism. EMATs are noncontact transducers, so they are promising in many situations where traditional piezoelectric transducers are not convenient, such as testing hot or moving samples. Composed of magnets and coils, they are also very flexible and capable of generating various kinds of waves such as the bulk waves and many types of guided waves (Thompson, 1973, 1979; Edwards et al., 2006).

Despite the obvious advantages, the energy transduction efficiencies of EMATs are relatively low compared to their piezoelectric counterparts (Hirao and Ogi, 2003). Often the signal of EMATs is at the level of several microvolts. With this situation, one imperative problem is to study the mechanism of the transducers further and build accurate theoretical models, and then try to design them optimally based on the models. The operations of EMATs are multiphysical in nature, and sometimes even nonlinearity exists, so their modeling is always a difficult task.

Here we mainly focus on the numerical models. Transient analysis of a meander-coil EMAT placed on isotropic nonferromagnetic half-space, assuming uniform static magnetic field was conducted in Ludwig and Dai (1991). The controlling eddy-current equations were studied in detail in Jafari-Shapoorabadi et al. (2001) and it was argued that the previous work using the total current divided by the cross section area of the conductor as the source current density was equivalently applying the incomplete equation, and this meant ignoring the skin effect and proximity effect, while we proved the opposite in Wang et al. (2016a) via customizing the underlying integrodifferential or normal differential equations. The finite element method (FEM) package COMSOL was used to build the electromagnetic model of a meander EMAT, and the simulated Lorentz force was exported to another package, Abaqus, as the driving force to excite Lamb waves in Dhayalan and Balasubramaniam (2010). The above-mentioned modeling work only involves nonmagnetic materials. There is also some initial work on modeling EMATs used to test magnetic material, which we will not discuss further here.

With proper models of EMATs, the next step is to optimize their design so as to maximize the testing performance. The work on optimizations of EMATs is still rare. A parametric study of an EMAT composed of a racetrack coil was conducted in Mirkhani et al. (2004) by varying the ratio of the width of the magnet to the width of the coil, and it was found that if this ratio was set at 1.2, the amplitude of the ultrasonic beam would be improved. One design variable and one objective function were used in this optimization, accomplished only through observation of a set of curves corresponding to different design variables, instead of using a real optimization algorithm. A spiral coil EMAT was optimized using genetic algorithm optimization procedure in the global optimization toolbox of Matlab in Seher et al. (2014, 2015). The ratio of the amplitude of the A0-mode Lamb waves to that of the S0 mode was selected as the objective function to be maximized, i.e., preferably generating the A0 mode.

In the evolutionary optimization algorithms, a complex numerical model is evaluated a great number of times, which is very time-consuming. In order to resolve this problem, an alternative mathematical model could be built to approximate the original computationally expensive numerical model, i.e., the evaluations of the original model could be replaced by the evaluations of the computationally less expensive approximate model, called a metamodel, surrogate model, etc. Implemented with polynomials, kriging, radial basis functions, and so on, the metamodels are already applied in electromagnetic device design and optimization (Sykulski, 2008), but seldom found in the field of ultrasonic NDT, or optimization of EMATs, so we explore one type of metamodel, the artificial neural networks here in the context of EMAT design and optimization. Obviously neural networks are used widely in many different situations, and being used as a metamodel is only one of their applications.

In this work we focus on the artificial neural networks as function approximators, or as metamodels of computationally expensive numerical models. Multilayer feedforward networks and radial basis function networks are both considered. Their performances for function approximations are tested on an example problem, with programs developed by the authors. Then modeling of an EMAT with COMSOL is discussed briefly, and methods to calculate the amplitudes of the displacement components of the generated Lamb waves are introduced. Finally the neural networks are applied as metamodels in the optimizations of the EMAT to accelerate the whole process.

2 Artificial neural networks as function approximators

There are many parameters in the finite-element model of an EMAT, such as the geometrical parameters, the magnitude and number of periods of the tone-burst waveform of the current excitation signal, the strength of the magnet, the material parameters, and so on. We can select a subset of the parameters as the inputs of a forward model. Similarly, with the FEM model, different quantities could be solved, including the spatial distributions of the magnetic, eddy current, force and displacement fields at some time instants; the waveforms of the displacement, stress and strain components at a point; some extracted features of the waveforms; etc. Some of the solved quantities can serve as the outputs of the forward model. Then this forward model is used in the optimizations in an iterative way. The forward model is a black-box function mapping the inputs (design variables in the optimizations) to the outputs (objective functions in the optimizations). Evaluations of the black-box function are of-

ten computationally expensive. Considering this, we want to approximate the function or forward model and substitute the approximation for the real forward model in the optimizations, where a great number of evaluations of the function are required.

Artificial neural networks are widely used in different fields. We are interested here in their abilities to approximate computationally expensive functions or the forward models of EMATs. Two different kinds of neural networks are considered. The first kind is the multilayer feedforward network, and the second kind is the radial basis function network. These two networks have different structures and different approaches to implementing approximations of functions. We review important concepts and algorithms of these networks and test them on a mathematical function. All the related code were implemented by the authors in Matlab. In the descriptions of the networks, vectors and matrices are used whenever possible to simplify the expressions.

2.1 Multilayer feedforward network

The structure of a two-layer feedforward neural network is shown in Fig. 1. There are N^1 neurons in the first (hidden) layer and N^2 neurons in the second (output) layer. Superscript is used to indicate the number of the layer. The vector $\boldsymbol{x}$ is the input column vector with N^I elements, and $\boldsymbol{y}$ is the corresponding output column vector with $N^O = N^2$ elements. In layer 1, $\mathbf{W}^1$ is the $N^1 \times N^I$ weight matrix, arranged such that the elements of the ith row correspond to the weights of the input elements $x_1, \ldots, x_{N^I}$ into the ith neuron. Then for the column vector $\mathbf{W}^1\boldsymbol{x}$, each element is a linear combination of $x_1, \ldots, x_{N^I}$. The term $\boldsymbol{b}^1$ is the bias column vector with one element for each neuron. The term $\boldsymbol{n}^1 = \mathbf{W}^1\boldsymbol{x} + \boldsymbol{b}^1$ is the net input to the transfer function $\boldsymbol{f}^1$ of layer 1, of which the output $\boldsymbol{a}^1 = \boldsymbol{f}^1(\boldsymbol{n}^1)$ is also the input to layer 2. Similar equations exist for layer 2, i.e., the final output is $\boldsymbol{y} = \boldsymbol{f}^2(\boldsymbol{n}^2)$, in which the net input is $\boldsymbol{n}^2 = \mathbf{W}^2\boldsymbol{a}^1 + \boldsymbol{b}^2$. Bold font is used for $\boldsymbol{f}^1$ and $\boldsymbol{f}^2$ to stress that these functions accept a vector and generate a vector. Operating on one element of the net input vector, the scalar transfer function f can take the following typical forms (Hagan et al., 2014).

$$\begin{aligned}
&\text{log-sigmoid: } f(n) = \frac{1}{1+e^{-n}}\\
&\text{tangent sigmoid: } f(n) = \frac{e^{-n}-e^{-n}}{e^{n}+e^{-n}}\\
&\text{linear: } f(n) = n
\end{aligned} \tag{1}$$

For the purpose of function approximation, we use the two-layer network in Fig. 1. The transfer function of the first layer could be the log-sigmoid function or the tangent-sigmoid function, while the transfer function of the second layer is the linear function. N^I is the number of input variables, and $N^2 = N^O$ is the number of output variables. Next

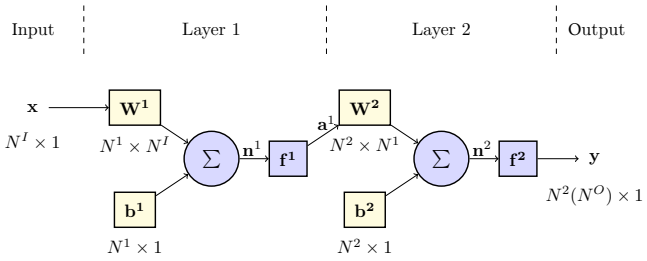


Figure 1. Multilayer feedforward neural network.

we must choose N^1, the number of neurons in the hidden layer, and apply some training algorithm to compute the best possible $\mathbf{W}^1$, $\boldsymbol{b}^1$, $\mathbf{W}^2$ and $\boldsymbol{b}^2$.

2.1.1 Basic back propagation

A widely used training algorithm for the multilayer neural network is the back propagation algorithm. Suppose the set of known input vectors is $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q\}$, and the corresponding set of known target vectors is $\{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_Q\}$. In iteration k of the training process, one randomly selected input vector $\boldsymbol{x}(k)$ is presented to the network, so the corresponding error vector is $\boldsymbol{e}(k) = \boldsymbol{t}(k) - \boldsymbol{y}(k)$. The performance index to minimize for the current iteration is

$$P = \boldsymbol{e}(k)^T\boldsymbol{e}(k). \tag{2}$$

According to the steepest descendent algorithm, the updating equations for the weight matrix and bias vector of layer m is

$$\begin{aligned}
\mathbf{W}^m(k+1) &= \mathbf{W}^m(k) - \alpha\boldsymbol{s}^m\left(\boldsymbol{a}^{m-1}\right)^T\\
\boldsymbol{b}^m(k+1) &= \boldsymbol{b}^m(k) - \alpha\boldsymbol{s}^m,
\end{aligned} \tag{3}$$

in which α is the learning rate. Here we define $\boldsymbol{a}^0 = \boldsymbol{x}$ in Fig. 1. $\boldsymbol{s}^m$ is the vector of sensitivity with element $s_i^m = \frac{\partial P}{\partial n_i^m}$, i.e., derivative of the performance index P with respect to the ith net input of layer m. The sensitivity of every layer is solved according to the recurrence equation,

$$\boldsymbol{s}^m = \dot{\mathbf{F}}^m\left(\mathbf{W}^{m+1}\right)^T\boldsymbol{s}^{m+1}, \tag{4}$$

in which,

$$\dot{\mathbf{F}}^m = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \ldots & 0\\ 0 & \dot{f}^m(n_2^m) & \ldots & 0\\ \ldots & \ldots & \ldots & \ldots\\ 0 & 0 & \ldots & \dot{f}^m(n_{N^m}^m) \end{bmatrix}, \tag{5}$$

where f^m is the scalar transfer function of layer m.

From Eq. (4), the sensitivity vectors of previous layers could be calculated from the sensitivity vector of the last

layer recurrently, from which the name "back propagation" derives.

As the starting point for the equation, the sensitivity vector for the last layer (layer M) is

$$\boldsymbol{s}^M = -2\dot{\mathbf{F}}^M(\boldsymbol{t}-\boldsymbol{y}). \tag{6}$$

To evaluate the capability of the two-layer network, trained with the basic back propagation algorithm, to approximate functions, we test the developed program on the following problem (Branin function) with two inputs and one output.

$$f(x_1,x_2) = 5x_1 + 10\left[1+\left(1-\frac{1}{8\pi}\right)\cos(15x_1-5)\right] + \left[-\frac{127.5}{4\pi^2}(3x_1-1)^2 + \frac{25}{\pi}(3x_1-1) + 15x_2 - 6\right]^2$$
$$x_1, x_2 \in [0,1] \tag{7}$$

The contour plot of this function is shown in Fig. 2a.

Figure 2b is the contour plot of the approximation by the multilayer neural network trained with the basic back propagation algorithm. The training data are sampled on a 30×30 grid of the two input variables. The transfer function of the first layer is the log-sigmoid function. A total of 60 neurons are used in the hidden layer, and 4×10^6 iterations are applied. The learning rate is $\alpha = 0.005$. The original function in Fig. 2a and the approximation in Fig. 2b are both generated on a 60×60 grid.

2.1.2 Improved back propagation

The basic back propagation algorithm is based on the steepest-descendent algorithm. Newton's method could be applied to achieve improvement on the basic algorithm, called the Levenberg–Marquardt (LM) back propagation (Hagan et al., 2014). All the training data are presented to the network at the same time. The error vector corresponding to the qth input $\boldsymbol{x}_q$ is $\boldsymbol{e}_q = \boldsymbol{t}_q - \boldsymbol{y}_q$. The components of Q error vectors are arranged into one column vector,

$$\boldsymbol{e} = \left(e_{1,1},\ldots,e_{N^M,1},e_{1,2},\ldots,e_{N^M,Q}\right)^T. \tag{8}$$

For element $e_{i,j}$, subscript i represents position in the error vector $\boldsymbol{e}_j$. The total number of error components in $\boldsymbol{e}$ is $N^e = Q \times N^M$.

The performance index to minimize is

$$P = \boldsymbol{e}(k)^T\boldsymbol{e}(k), \tag{9}$$

in which k is the number of iteration.

The unknown variables to solve, including the weights and the biases, are also arranged into one column vector:

$$\boldsymbol{v} = \left(w^1_{1,1},\ldots,w^1_{1,N^I},w^1_{2,1},\ldots,w^1_{2,N^I},\ldots,w^1_{N^1,N^I}, b^1_1,\ldots,b^1_{N^1},w^2_{1,1},\ldots,b^M_{N^M}\right)^T. \tag{10}$$

The total number of variables is $N^v = N^1(N^I+1)+N^2(N^1+1)+\ldots+N^M(N^{M-1}+1)$.

The Jacobian matrix is defined as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_1}{\partial v_1} & \frac{\partial e_1}{\partial v_2} & \cdots & \frac{\partial e_1}{\partial v_{N^v}} \\ \frac{\partial e_2}{\partial v_1} & \frac{\partial e_2}{\partial v_2} & \cdots & \frac{\partial e_2}{\partial v_{N^v}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{N^e}}{\partial v_1} & \frac{\partial e_{N^e}}{\partial v_2} & \cdots & \frac{\partial e_{N^e}}{\partial v_{N^v}} \end{bmatrix}. \tag{11}$$

The updating equation for the unknown variables is

$$\boldsymbol{v}(k+1) = \boldsymbol{v}(k) - \left(\mathbf{J}^T\mathbf{J}+\mu\mathbf{I}\right)^{-1}\mathbf{J}^T\boldsymbol{e}, \tag{12}$$

in which μ is a small value added to the diagonal elements of $\mathbf{J}^T\mathbf{J}$ to ensure that it is invertible.

The most difficult part is calculating the $\mathbf{J}$ matrix. The elements of $\mathbf{J}$ could be solved with sensitivity back propagation just like in the basic back propagation algorithm. The processing here is more complex because there are N^e sensitivity vectors to propagate, instead of one sensitivity vector per iteration in the basic algorithm. This training algorithm of the two-layer network is also implemented and tested on the example problem.

The result of approximation of the test function with the multilayer network trained with the LM algorithm is shown in Fig. 2c. A total of 60 neurons are used in the hidden layer, and 800 iterations are applied.

2.2 Radial basis function network

Another candidate neural network for function approximation is the radial basis function neural network (RBFNN). The basic structure of RBFNN is shown in Fig. 3. There are N^C neurons in the first (radial basis function, hidden) layer, and N^2 neurons in the second (linear, output) layer. In the RBFNN, the matrix $\mathbf{C}$ no longer contains weights applied to the elements of the input vector. Instead, the distance (represented with $||\cdot||$) between the input vector and each row of the matrix $\mathbf{C}$ is computed to generate a $N^C \times 1$ column vector, which is then multiplied element by element (represented with $.\times$) with the scale vector $\boldsymbol{s}$.

The transfer function of the radial basis function layer is Gauss function,

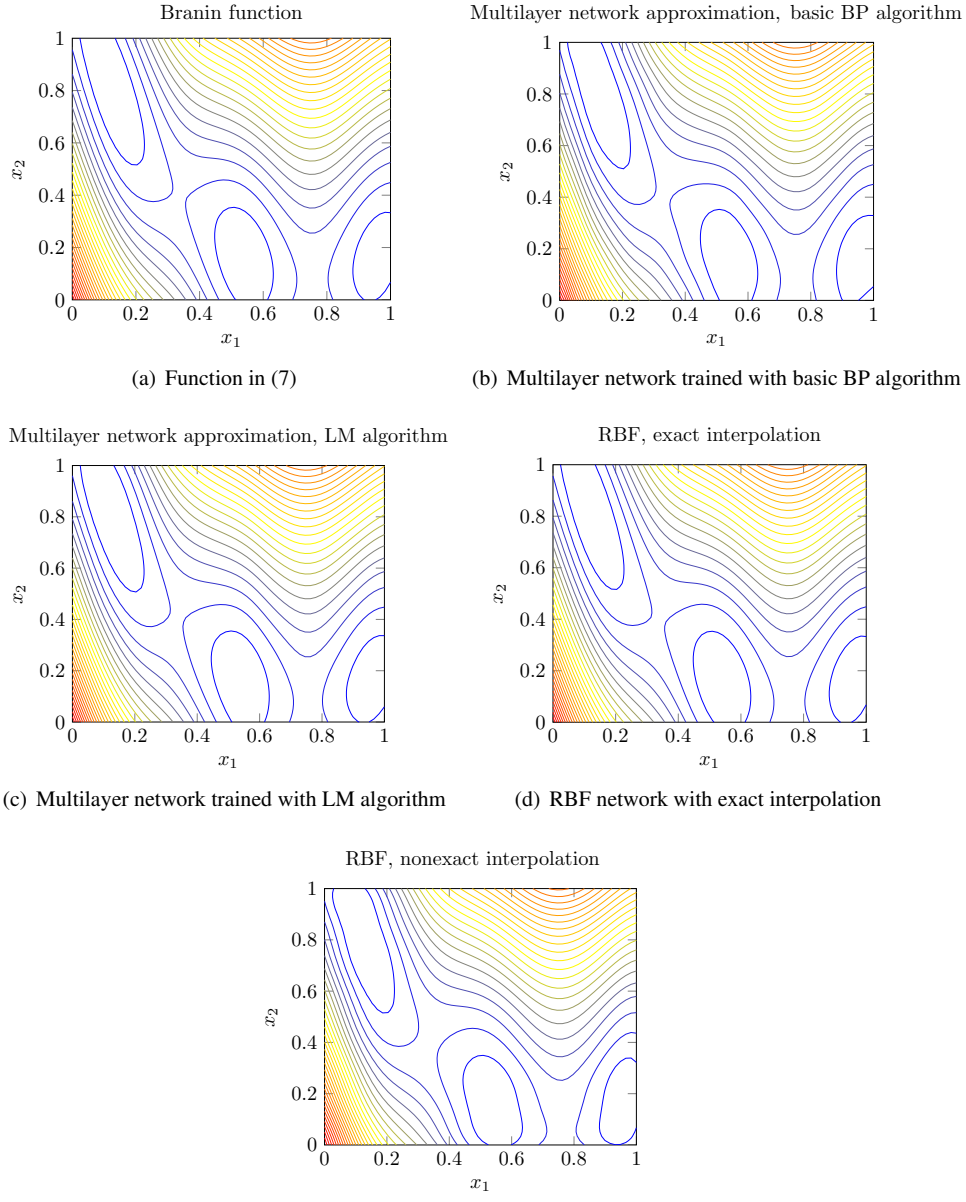
$$\text{Gauss}: f(n) = e^{-n^2}, \tag{13}$$

which is a bell-shaped function concentrated at $n = 0$. Then for the first RBFd layer, we have

$$a^1_i = f^1(s_i||\boldsymbol{x}-\boldsymbol{c}_i||), \tag{14}$$

in which $\boldsymbol{c}_i$ is the transposition of the ith row of $\mathbf{C}$, i.e.,

$$\mathbf{C} = \begin{bmatrix} \boldsymbol{c}_1^T \\ \cdots \\ \boldsymbol{c}_{N^C}^T \end{bmatrix}. \tag{15}$$



(a) Function in (7)

(b) Multilayer network trained with basic BP algorithm

(c) Multilayer network trained with LM algorithm

(d) RBF network with exact interpolation

(e) RBF network with nonexact interpolation

Figure 2. Test function and the approximations.

The transfer function of the second layer is the linear function defined in Eq. (1), so element k of the output vector $\boldsymbol{y}$ is

$$y_k = w_{k,1}a_1^1 + \ldots + w_{k,N^C}a_{N^C}^1 + b_k. \tag{16}$$

Since the transfer function f^1 is a local function around scalar 0, equation Eq. (16) states that the output component y_k is a linear combination of N^C local functions, each concentrated around $\boldsymbol{c}_i$ and scaled by s_i, and finally biased by b_k. The distance operator $||\cdot||$ makes the output of the transfer function f^1 symmetric about center vector $\boldsymbol{c}_i$, from which the name "radial basis function" derives.

Training of the RBFNN is quite different from that of the general multilayer networks. Normally the process is divided into two stages. In the first stage, the centers (rows of $\mathbf{C}$) and scales (elements of $\boldsymbol{s}$) are selected according to some criterion. Generally the centers should be distributed evenly in the input space. The scales should be set so that the adjacent basis functions overlap somewhat with each other. In the programs developed for this work, a spread parameter is used to specify the scale as $s = \frac{\sqrt{-\ln\frac{1}{2}}}{\text{spread}}$, so that if the distance is spread, the basis function drops to half the maximum

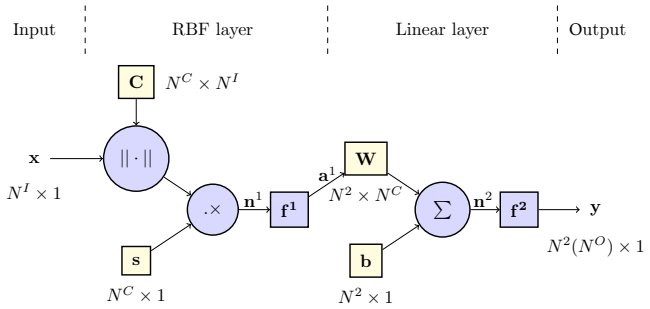


Figure 3. Radial basis function neural network.

value. In the second stage, $\mathbf{C}$ and $\boldsymbol{s}$ are fixed, and $\mathbf{W}$ and $\boldsymbol{b}$ are solved.

2.2.1 Exact interpolation

One common scheme is to select the centers of the radial basis functions as the input vectors. Then the number of centers N^{C} will be equal to the number of the known input vectors Q. For now we focus on one output component, y_k. If we let the function approximation pass through the known input/output data, assuming $\boldsymbol{b} = 0$,

$$\begin{cases} w_{k,1} f^1(s_1 \|\boldsymbol{x}_1 - \boldsymbol{x}_1\|) + \ldots + w_{k,Q} f^1(s_Q \|\boldsymbol{x}_1 - \boldsymbol{x}_Q\|) & = & y_{k,1} \\ \cdots & & \cdots \\ w_{k,1} f^1(s_1 \|\boldsymbol{x}_Q - \boldsymbol{x}_1\|) + \ldots + w_{k,Q} f^1(s_Q \|\boldsymbol{x}_Q - \boldsymbol{x}_Q\|) & = & y_{k,Q} \end{cases}, \quad (17)$$

in which the second subscript of y is the index of the input vectors.

In matrix form,

$$\mathbf{R}\boldsymbol{w}_k = \boldsymbol{y}_k, \quad (18)$$

in which the element of matrix $\mathbf{R}$ is $R_{i,j} = f^1(s_j \|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$, $\boldsymbol{w}_k = (w_{k,1}, \ldots, w_{k,Q})^T$, and $\boldsymbol{y}_k = (y_{k,1}, \ldots, y_{k,Q})^T$.

In the system of equations Eq. (17), the number of unknown weights and the number of equations are equal, so the weights could be solved with matrix inversion as $\boldsymbol{w}_k = \mathbf{R}^{-1}\boldsymbol{y}_k$. An exact interpolation is obtained, because the approximation passes all the known data pairs. If there are multiple outputs ($N^O > 1$), the previous process could be repeated for every output component in the set $\{y_1, y_2, \ldots, y_{N^O}\}$.

The disadvantage of the approach of exact interpolation is that we need as many radial basis functions as the number of the input vectors, so there will be too many basis functions or neurons in the hidden layer when many input vectors are available. Another problem is that if the data contain noise, the exact interpolation will lead to overfitting. This is better demonstrated with a simple problem. Figure 4 shows the function $y = \sin(x), x \in [0, 2\pi]$, its samples with noise (level of 0.1), exact interpolation with RBFNN and the nonexact interpolation.

The result of approximation of the test function with the RBF network with exact interpolation is shown in Fig. 2d.

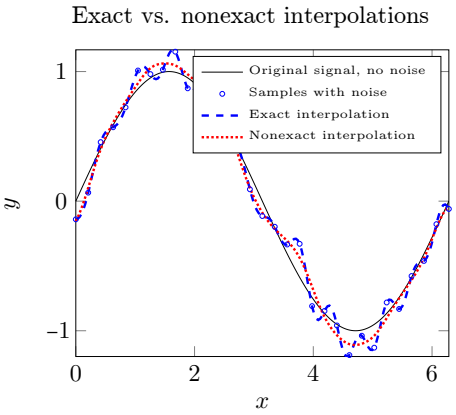


Figure 4. Exact and nonexact interpolations of a sine signal with noise.

The input vectors are generated on the 30×30 grid, and they are also used as the centers of the radial basis functions. The spread parameter is 0.5, so that the value of the basis function drops to $1/2$ of the maximum value at a distance of 0.5 from the center.

In the process of structural optimization, the geometry of the EMAT model is continuously changing, so that the mesh of the model or, more specifically, the number of the elements, the coordinates of the nodes, etc. are also changing. In this way a noise or random error exists in the optimization process. For this reason, it is necessary to check the nonexact interpolation and compare the result with that of the exact interpolation to see the effect of meshing noise.

2.2.2 Nonexact interpolation

With the disadvantages of the version of the RBF network using exact interpolation, one modification is to use fewer radial basis functions in the hidden layer. The centers of the basis functions are not required to be in the set of the known input vectors.

We focus on one output component y_k. The performance index is

$$P = \sum_{q=1}^{Q} e_q^2, \quad (19)$$

in which $e_q = t_{k,q} - y_{k,q}$.

The augmented vector of unknown variables is

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{w}_k \\ b_k \end{bmatrix}, \quad (20)$$

in which $\boldsymbol{w}_k$ is transposition of the kth row of $\mathbf{W}$ as

$$\mathbf{W} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \cdots \\ \boldsymbol{w}_{N^O}^T \end{bmatrix}. \quad (21)$$

For input vector $\boldsymbol{x}_q$, the augmented input to the second layer is

$$\boldsymbol{a}_q = \begin{bmatrix} \boldsymbol{a}_q^1 \\ 1 \end{bmatrix}. \tag{22}$$

Next we define

$$\mathbf{U} = \begin{bmatrix} \boldsymbol{a}_1^T \\ \cdots \\ \boldsymbol{a}_Q^T \end{bmatrix}. \tag{23}$$

Then the vector of weights and bias of the second output layer is solved as

$$\boldsymbol{v} = \left(\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right)^{-1}\mathbf{U}^T\boldsymbol{t}, \tag{24}$$

in which

$$\boldsymbol{t} = \begin{bmatrix} t_{k,1} \\ \cdots \\ t_{k,Q} \end{bmatrix}. \tag{25}$$

The term ρ denotes a small value added to diagonal elements of $\mathbf{U}^T\mathbf{U}$ to prevent overfitting.

The result of approximation of the test function with the RBF network with nonexact interpolation is shown in Fig. 2e. The input vectors are generated on the 30×30 grid. The centers of the radial basis functions are generated on the 30×30 grid. The spread parameter is 0.2.

3 Time- and frequency-domain modeling of an omnidirectional EMAT

We consider an omnidirectional EMAT composed of a spiral coil and a cylindrical permanent magnet, used to generate omnidirectional Lamb waves in a plate.

The coil is composed of tightly wound copper wires, instead of forming a meander pattern, so both S0-mode and A0-mode Lamb waves will be generated. In this work, the aim is to preferably generate A0-mode Lamb waves, so we built the model bearing this preference in mind. Justification of this choice of wave mode could be found in Huthwaite et al. (2013), where it was explained that the thickness variations were more sensitive with the A0 wave mode.

In this section, the formulations of an axisymmetric EMAT are given first (further details can be found in Wang et al., 2016b), and then the numerical model is described.

3.1 Formulations of an axisymmetric EMAT

The basic equations for the electromagnetic field simulation in the study of EMATs are Maxwell's equations (Ida, 2007).

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{26}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \tag{27}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{28}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{29}$$

These equations are Faraday's law, Ampere–Maxwell's law, Gauss's law for electric fields and Gauss's law for magnetic fields, respectively. $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic flux density, t is the time variable, $\mathbf{H}$ is the magnetic field strength, $\mathbf{J}$ is the current density, $\mathbf{D}$ is the electric flux density, and ρ is the charge density. The term $\nabla\times$ denotes curl of vector, and $\nabla\cdot$ is divergence of vector. Since the frequency in normal EMAT operation is no higher than several MHz, the term $\frac{\partial \mathbf{D}}{\partial t}$ in Ampere–Maxwell's law could be neglected.

To solve Maxwell's equations, another set of equations called the constitutive equations is needed:

$$\mathbf{B} = \mu\ \mathbf{H}, \tag{30}$$

$$\mathbf{D} = \epsilon\ \mathbf{E}, \tag{31}$$

in which μ is the magnetic permeability, and ϵ is the dielectric constant.

Because the magnetic flux density is solenoidal, magnetic vector potential (MVP) $\mathbf{A}$ is introduced through

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{32}$$

The following Columbo gauge is applied to define the MVP completely:

$$\nabla \cdot \mathbf{A} = 0. \tag{33}$$

With the MVP, the equation describing the eddy-current phenomenon is

$$-\frac{1}{\mu}\nabla^2\mathbf{A} + \sigma\frac{\partial \mathbf{A}}{\partial t} = \mathbf{J}_s, \tag{34}$$

in which ∇^2 is the vector Laplacian operator, σ is the conductivity, and $\mathbf{J}_s$ is the source current density. The eddy-current density, not written explicitly in Eq. (34), is $\mathbf{J}_e = -\sigma\frac{\partial \mathbf{A}}{\partial t}$. This equation holds where there is a conductor and a source current flows inside of this conductor. At the region without the source current, the $\mathbf{J}_s$ term is dropped. Where there is no conductor, like in the air, the negative eddy-current density term $\sigma\frac{\partial \mathbf{A}}{\partial t}$ (or $-\mathbf{J}_e$) is also dropped. The magnetic permeability μ appears in the denominator in Eq. (34), which implies that the material is isotropic. Besides this, an additional assumption in this work is that the material is linear, so that μ is fixed (uniformly distributed) in every material of the model.

For 2-D axisymmetric field simplification assuming $\mathbf{J}_s$ is perpendicular to the r–z plane, we have $\mathbf{J}_s = J_{s\phi}\mathbf{e}_\phi$, with $J_{s\phi}$ as the ϕ component of vector $\mathbf{J}_s$, and $\mathbf{e}_\phi$ as the unit vector along the ϕ axis. We also have $\mathbf{A} = A_\phi \mathbf{e}_\phi$. The term $\nabla^2\mathbf{A}$ (the vector Laplacian operator applied on vector $\mathbf{A}$) has to be treated carefully because, unlike in the Cartesian coordinates, we have in cylindrical coordinates

$$\nabla^2\mathbf{A} \neq \nabla^2 A_r \mathbf{e}_r + \nabla^2 A_\phi \mathbf{e}_\phi + \nabla^2 A_z \mathbf{e}_z. \tag{35}$$

Instead, the ϕ component of $\nabla^2\mathbf{A}$ is

$$(\nabla^2\mathbf{A})_\phi = \nabla^2 A_\phi + \frac{2}{r^2}\frac{\partial A_r}{\partial \phi} - \frac{A_\phi}{r^2}, \tag{36}$$

in which the scalar Laplacian operator applied on scalar A_ϕ is

$$\nabla^2 A_\phi = \frac{\partial^2 A_\phi}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 A_\phi}{\partial \phi^2} + \frac{\partial^2 A_\phi}{\partial z^2} + \frac{1}{r}\frac{\partial A_\phi}{\partial r}. \tag{37}$$

In a 2-D axisymmetric model, only the A_ϕ component is nonzero, and it is independent of ϕ:

$$A_\phi = A_\phi(r, z). \tag{38}$$

Then the ϕ component of $\nabla^2\mathbf{A}$ is simplified as

$$(\nabla^2\mathbf{A})_\phi = \frac{\partial^2 A_\phi}{\partial r^2} + \frac{\partial^2 A_\phi}{\partial z^2} + \frac{1}{r}\frac{\partial A_\phi}{\partial r} - \frac{A_\phi}{r^2}. \tag{39}$$

From now on, A_ϕ will be written as A and $J_{s\phi}$ be written as J_s for simplicity. With this notation, the vector Eq. (34) is transformed to the ϕ component scalar equation:

$$-\frac{1}{\mu}\left(\frac{\partial^2 A}{\partial r^2} + \frac{\partial^2 A}{\partial z^2} + \frac{1}{r}\frac{\partial A}{\partial r} - \frac{A}{r^2}\right) + \sigma\frac{\partial A}{\partial t} = J_s. \tag{40}$$

This equation is a diffusion equation describing the eddy-current phenomenon in cylindrical coordinates. Now the eddy-current density term is $J_e = -\sigma\frac{\partial A}{\partial t}$.

Next, it is crucial to correctly decide the source current density term J_s. One inaccurate way is to define total current divided by cross section area of the source conductor as the source current density. This definition was borrowed from magnetostatic simulation of magnetic field generated by a steady electric current. In alternating current simulation, the externally applied total current i should be the integral of the sum of the source current density J_s and the eddy-current density J_e,

$$i = \iint_S (J_s + J_e)\mathrm{d}S, \tag{41}$$

in which S is the cross section of the source conductor. Unlike in the 2-D planar model, J_s is no longer uniformly distributed in the cross section of the source conductor in cylindrical coordinates; instead, the product of J_s and r is a constant for a particular conductor, i.e., J_s is inversely proportional to r:

$$J_s r = C, \tag{42}$$

where C is a constant with a different value for each source conductor. Combined with Eq. (41), the constant C could be derived as

$$C = \frac{i + \iint_S \sigma\frac{\partial A}{\partial t}\mathrm{d}S}{\iint_S \frac{1}{r}\mathrm{d}S}, \tag{43}$$

so the original eddy-current equation Eq. (40) in cylindrical coordinates could now be written as

$$-\frac{1}{\mu}\left(\frac{\partial^2 A}{\partial r^2} + \frac{\partial^2 A}{\partial z^2} + \frac{1}{r}\frac{\partial A}{\partial r} - \frac{A}{r^2}\right) + \sigma\frac{\partial A}{\partial t} = \frac{i + \iint_S \sigma\frac{\partial A}{\partial t}\mathrm{d}S}{r\iint_S \frac{1}{r}\mathrm{d}S}. \tag{44}$$

For steady-state or frequency-domain analysis, the phasor notation is adopted:

$$-\frac{1}{\mu}\left(\frac{\partial^2 \dot{A}}{\partial r^2} + \frac{\partial^2 \dot{A}}{\partial z^2} + \frac{1}{r}\frac{\partial \dot{A}}{\partial r} - \frac{\dot{A}}{r^2}\right) + j\omega\sigma\dot{A} = \frac{\dot{i} + j\omega\sigma\iint_S \dot{A}\mathrm{d}S}{r\iint_S \frac{1}{r}\mathrm{d}S}, \tag{45}$$

in which the dots on A and i indicate they are complex phasors, ω is the angular frequency, and j is the imaginary unit. This steady-state equation was proposed in Preis (1983), following the work on the similar steady-state integrodifferential equation in a 2-D planar model in Konrad (1981).

In Wang et al. (2016b), we proposed solving the above integrodifferential equations in time and frequency domains via customizing the underlying equations in COMSOL package.

The model of an EMAT is multiphysics in nature. Besides the above electromagnetic equations, the equations describing the generation and propagation of the ultrasonic waves in an elastic solid are as follows (Auld, 1990).

$$\nabla\cdot\mathbf{T} = \rho'\frac{\partial^2\mathbf{u}}{\partial t^2} - \mathbf{F} \tag{46}$$

$$\mathbf{T} = \mathbf{c} : \mathbf{S} \tag{47}$$

$$\mathbf{S} = \nabla_s\mathbf{u} \tag{48}$$

These equations are equation of motion, Hook's law and strain-displacement relation, respectively. $\mathbf{T}$ is the stress tensor, ρ' is the density (symbol $'$ is used to differentiate it from charge density in Maxwell's equations), $\mathbf{F}$ is the body force, $\mathbf{c}$ is the stiffness tensor, $\mathbf{S}$ is the strain tensor, and $\mathbf{u}$ is the displacement vector. The symbol : is the double dot product of a fourth rank tensor $\mathbf{c}$ and a second rank tensor $\mathbf{S}$, and $\nabla_s\mathbf{u}$ is the symmetric part of the gradient of the vector $\mathbf{u}$.

For homogenous and isotropic media, from the above elastodynamic equations, Navier's equation could be derived as

$$\mu'\nabla^2\mathbf{u} + (\lambda + \mu')\nabla(\nabla\cdot\mathbf{u}) = \rho'\frac{\partial^2\mathbf{u}}{\partial t^2} - \mathbf{F}. \tag{49}$$

Here, λ and μ' are Lamé constants. The symbols $'$ in μ' is used to differentiate it from the magnetic permeability.

The link between the electromagnetic equations and the elastodynamic equations is the Lorentz force defined as

$$\mathbf{F}_L = \mathbf{J} \times \mathbf{B} = \mathbf{J} \times (\mathbf{B}_0 + \mathbf{B}_d), \tag{50}$$

in which $\mathbf{B}$ is the total magnetic flux density composed of the static flux density $\mathbf{B}_0$ of the bias magnet and the dynamic flux density $\mathbf{B}_d$ generated by the excitation coil. For moderate magnitude exciting current, the $\mathbf{B}_d$ term is usually very small compared to $\mathbf{B}_0$.

3.2 The omnidirectional EMAT model

The complete EMAT model is composed of one magnetostatic submodel describing the magnetic field of the permanent magnet, one eddy-current submodel analyzing the eddy-current phenomenon accompanied by the skin and proximity effects, and one elastodynamic submodel for the simulation of generation and propagation of Lamb waves in the plate. The two electromagnetic submodels share one geometry containing the air, the inner section of the plate, the copper wires and the permanent magnet, as in Fig. 5. Note that in this geometry only the inner section of the full plate is modeled. The elastodynamic submodel has its own geometry, only containing the full plate. The Lorentz forces calculated from the two electromagnetic submodels are transferred to the elastodynamic submodel as the driving forces of Lamb waves. With these two geometries, the structure of the EMAT model is very clear, compared to some previous work.

Additionally, we can use different meshing rules for these two geometries, according to the respective physics. This two-geometry treatment is valid because the Lorentz forces are confined in the region of the plate just under the transducer.

In Fig. 5, it is only necessary to consider the region where $r > 0$, since this is an axisymmetric model. The testing frequency is 50 kHz. As in Seher et al. (2014), the relative magnetic permeability of the steel plate is 160, i.e., it is a simplified linear material. The conductivity is $4.032\,\mathrm{MS\,m^{-1}}$. The thickness of the plate is 10 mm. The remanent magnetic flux density of the magnet is set to 1.3 T along the positive direction of the z axis. R_M is the radius of the magnet, and l_M is the liftoff distance of the magnet from its base to the top of the coil. The two parameters of R_M and l_M will be used as the design variables in the optimizations based on the genetic algorithm and the neural-network metamodels, while all the other parameters are fixed.

R_C is the average radius of the coil decided as

$$R_C = (2n-1)\frac{\lambda}{4}, n = 1, 2, \ldots, \tag{51}$$

in which λ is the wavelength of the desired Lamb wave mode. For the EMAT on the steel plate, n is chosen to be 1, i.e., $R_C = \frac{\lambda}{4}$, similar to Seher et al. (2015).

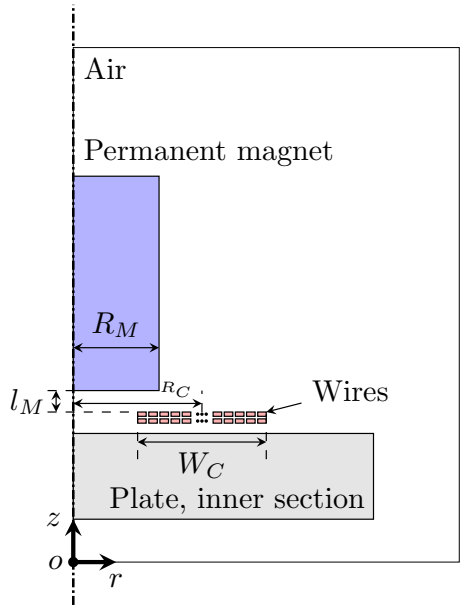


Figure 5. The geometry of the electromagnetic submodels. This geometry is used for magnetostatic analysis and eddy-current analysis.

As stated previously, we want to selectively generate A0-mode Lamb waves. For A0-mode Lamb waves in a steel plate of 10 mm thickness at 50 kHz, from the dispersion curves generated with a program we developed, the phase velocity is $1867.78\,\mathrm{m\,s^{-1}}$, and then the wavelength λ is 37.36 mm.

W_C is the radial width of the coil (difference between the outer and inner radii of the coil). The coil is composed of two layers of copper wires with conductivity as $5.998 \times 10^7\,\mathrm{S\,m^{-1}}$. The wires form an array of 23 columns and 2 rows, as in Wilcox et al. (2005). The wires have rectangular cross sections. The radial width of each wire is 0.3 mm, and the radial gap between adjacent wires in the same layer is 0.1 mm. The axial height of the wire and the gap between the two layers are both 0.1 mm. We have chosen to model each wire individually, so that all the formulations in Sect. 3.1 are applicable.

The geometry of the elastodynamic submodel simply contains a full plate with a radius of 1.2 m. Young's modulus is 200×10^9 Pa, Poisson's ratio is 0.33, and the density is $7850\,\mathrm{kg\,m^{-3}}$. The observation point to record the displacement components in the simulations is located at 60 cm from the z axis, in the middle plane of the plate. From the displacement wave structures of Lamb waves with the specified frequency and plate thickness, at the middle plane of the plate, the displacement component $u = u_r$ only corresponds to the S0 mode, while the other component $w = u_z$ only corresponds to the A0 mode.

The boundaries of the submodels must be handled with care. In the geometry for the electromagnetic submodels, there is a layer of infinite elements at the air boundary sim-

ulating an air region extending infinitely. In the geometry of the elastodynamic submodel, the top and bottom boundaries of the full plate are free boundaries without constraints or loads. For a transient time-domain analysis, the outer edge of the plate (at $r = 1.2$ m) is also a free boundary. If the full plate is long enough in the radial direction and the total time of simulation is limited to a proper value, the reflections from the end of the plate can be avoided. For a frequency-domain analysis, an extra perfectly matched layer (PML) must be added to the end of the plate so that the energy in the plate can dissipate.

To further increase the accuracy of the model, fillets are added to the sharp corners of the magnet and the wires, so that the geometrical singularities are removed, while at the same time the number of elements and hence the complexity of the model is also increased.

4 Evaluations of the amplitudes of the displacement components

In optimizations of the EMAT, the amplitudes of the displacement components will be used to calculate the objective functions, so we must decide how to evaluate the amplitudes. We explore three approaches as in the following subsections.

4.1 Amplitudes from time-domain simulations

A natural choice is to do time-domain simulations and record the time waveforms of the displacement components and then calculate the envelopes of the waveforms and solve the maximum values as the amplitudes. These time-dependent simulations require small time steps to ensure the convergence, and thus are very time-consuming. Since we will apply evolutionary optimization algorithms, a great number of evaluations of the objective functions will be needed, so this approach is not practical. Nevertheless the time waveforms will be simulated, serving as references to test the other approaches. In this work, the number of time steps is usually set as 6000, for the tone-burst excitation signal $x(t)$ composed of 5 sinusoidal periods modulated with a Hanning window function.

4.2 Amplitudes from frequency-domain simulations and the FFT/IFFT processing

The second approach is to transform the excitation time-domain burst signal into its spectrum with FFT, input them into the frequency-domain model of the EMAT, transform the resulting modified spectrum back into time domain with IFFT to obtain the time waveforms of the displacement components,

$$\begin{cases} u(t) &= \mathcal{F}^{-1}\{\mathcal{F}[x(t)]H_u(\omega, R_\mathrm{M}, l_\mathrm{M})\} \\ w(t) &= \mathcal{F}^{-1}\{\mathcal{F}[x(t)]H_w(\omega, R_\mathrm{M}, l_\mathrm{M})\}, \end{cases} \tag{52}$$

in which $\mathcal{F}$ represents the Fourier transform, $\mathcal{F}^{-1}$ is the inverse Fourier transform, $x(t)$ is the input tone-burst signal, $H_u(\omega, R_\mathrm{M}, l_\mathrm{M})$ is the frequency response for the displacement component u, and $H_w(\omega, R_\mathrm{M}, l_\mathrm{M})$ is the frequency response for the displacement component w. R_M and l_M are included to stress that these frequency responses change with the design variables, while the input signal $x(t)$ is fixed. From the converted time waveforms, the envelopes are calculated and the peaks of the envelopes are solved as the desired amplitudes.

The spectrum of the excitation signal is bell-shaped and concentrated around the center frequency, so we could focus on the part of spectrum which is bigger than some predefined threshold value. For example, this threshold might be defined as a percentage of the maximum value of the spectrum. Generally tens of or even several spectrum components are enough for this approach, so it is less time-consuming than the first approach where an excessive number of steps are required. As an example, u and w waveforms from the time-dependent simulation and the frequency-domain model with FFT/IFFT processing are compared in Fig. 6. The design variables are selected as $R_\mathrm{M} = 8$ mm and $l_\mathrm{M} = 1$ mm. The threshold to select the frequency components is 10 %, that is, only the frequency components higher than 10 % of the peak value of the spectrum are used, and others are discarded. With this threshold, 11 components around the center frequency are kept. From the figure, we can see that the time waveforms from the multifrequency model with FFT/IFFT processing are very close to the waveforms directly solved from the transient analysis. From various tests, we found that this is only possible when we model each wire of the coil individually. For other current sources, the waveforms from the two approaches are different.

One important condition to apply the frequency model is that the whole model must be linear. This is satisfied only when the excitation current is small so that the dynamic magnetic field generated by the coil could be ignored.

4.3 Amplitudes from frequency-domain simulations with one single frequency

Since a narrow banded signal with spectrum around its center frequency is used in the testing, another possibility is using one single frequency in the frequency-domain model, i.e., resorting to a steady-state solution of the EMAT model. The solved displacement components are complex phasors, so their absolute values are used as the amplitudes

$$\begin{cases} |\dot{u}| &= |H_u(\omega_\mathrm{c}, R_\mathrm{M}, l_\mathrm{M})| \\ |\dot{w}| &= |H_w(\omega_\mathrm{c}, R_\mathrm{M}, l_\mathrm{M})|, \end{cases} \tag{53}$$

in which ω_c is the center frequency in radian, $\dot{u}$ is the complex phasor of u, and $\dot{w}$ is the complex phasor of w. This approach, as an approximation of the second approach, is valid because the spectrum of the excitation is bell-shaped and concentrated around the center frequency ω_c, and the

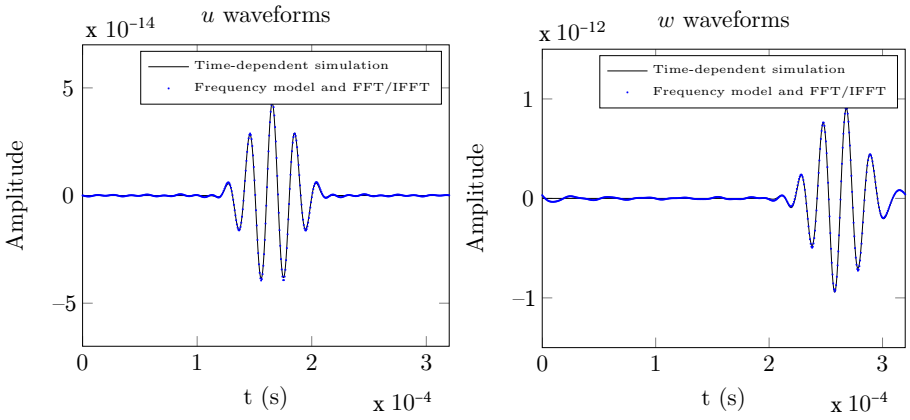


Figure 6. The u and w waveforms from time-dependent simulation and frequency-domain model with FFT/IFFT processing.

frequency responses H_u and H_w change slowly around the center frequency.

This method of evaluation of the amplitudes of the displacement components is the fastest. However, it should be stressed that this method is an approximation and so is not suitable when higher accuracy is desired.

5 Metamodels of the EMAT implemented with neural networks

With the time-domain and frequency-domain EMAT models, as well as the discussed approaches to obtain the amplitudes of the displacement components at the observation point, we can already calculate the objective functions of the optimization problem and solve the problem with some optimization algorithm. The approach using the frequency model with a single frequency seems the best because it is the fastest.

One problem is that the amplitudes from the single-frequency approach are only approximations. Now with the metamodels implemented with the artificial neural networks, we can resort to the more time-consuming multifrequency approach with FFT/IFFT processing, i.e., solving some sample points with the multifrequency approach, then building metamodels with these sample points, and finally employing these metamodels in the optimization algorithm.

With the design variables as the inputs, we are concerned with two outputs. The first is $\frac{A_u}{A_w}$, the ratio of the amplitude of u waveform to that of w waveform, and the second is A_w, the amplitude of w waveform. $\frac{A_u}{A_w}$ and $-A_w$ will be used as the two objective functions to minimize in the next section.

Since there are only two inputs, we can draw the surfaces of the outputs with respect to the two inputs. Figure 7 shows the $\frac{A_u}{A_w}$ and A_w surfaces solved with the multifrequency model with FFT/IFFT processing, on a 50×50 grid of the design variables. This figure and the corresponding set of data will be used as a reference to test the performances of the neural-network metamodels. On a computer installed with Intel Xeon CPU E5-2650 at 2.30 GHz and 128 GB RAM, running the Windows 10 operating system, the total time to solve the 2500 samples is 478 071.7 s. In the $\frac{A_u}{A_w}$ surface, there is a valley along the l_M axis, indicating that $\frac{A_u}{A_w}$ is mainly determined by the R_M value. Obviously the minimum value of $\frac{A_u}{A_w}$ should be in this valley.

Next we will build the neural-network metamodels based on a training set generated on a 25×25 grid of the design variables, i.e., 625 uniformly distributed samples will be used as the known training data. The time to calculate the samples of the training set is around 33 h.

First we explore the multilayer network with the basic BP algorithm. A total of 200 neurons are used in the hidden layer, the number of iterations is 6×10^7, and the learning rate is 0.02. The transfer function of the hidden layer is the log-sigmoid function. The transfer function of the output layer is the linear function. In one test run of the program, the training time for the $\frac{A_u}{A_w}$ function is 24 141.458 s, and the training time for the A_w function is 22 258.787 s. After the networks are built, the $\frac{A_u}{A_w}$ and A_w functions are predicted with the networks on a 60×60 grid of the design variables, as shown in Fig. 8. The values of $\frac{A_u}{A_w}$ and A_w are both scaled to $[-1, 1]$ before approximation with the network and the predicted values are scaled back accordingly to be drawn in the figure. From the figure it could be observed that the $\frac{A_u}{A_w}$ surface is smoother than it should be at the valley, with Fig. 7 as the reference, although the number of iterations is already big. From various tests with different parameters, we found that the multilayer network with the basic BP algorithm is difficult to train. With the error of approximation, the performance of this network might not be satisfactory.

Then we test the multilayer network with the LM algorithm. 200 neurons are used in the hidden layer, and the number of iterations is 2000. The transfer functions are the same as in the previous network trained with the BP algorithm. In one run of the program, the training time for the $\frac{A_u}{A_w}$ function is 1775.372 s, and the training time for the A_w function is 1763.376 s. After the networks are built, the $\frac{A_u}{A_w}$ and A_w

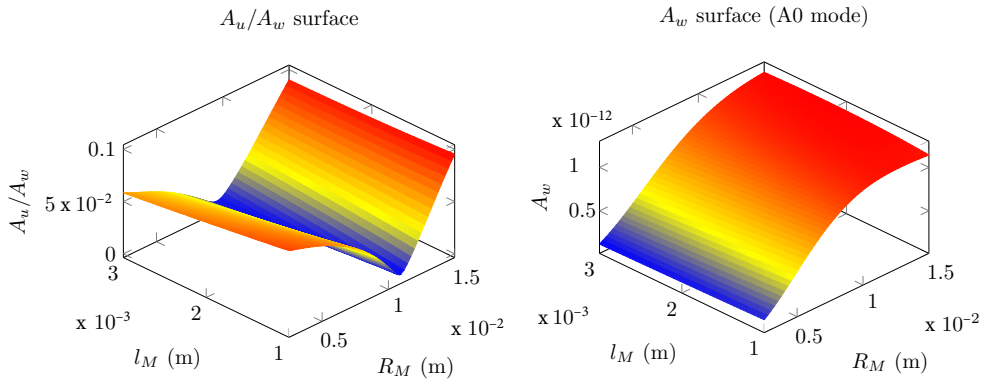


Figure 7. A_u/A_w and A_w surfaces from multifrequency model with FFT/IFFT processing.

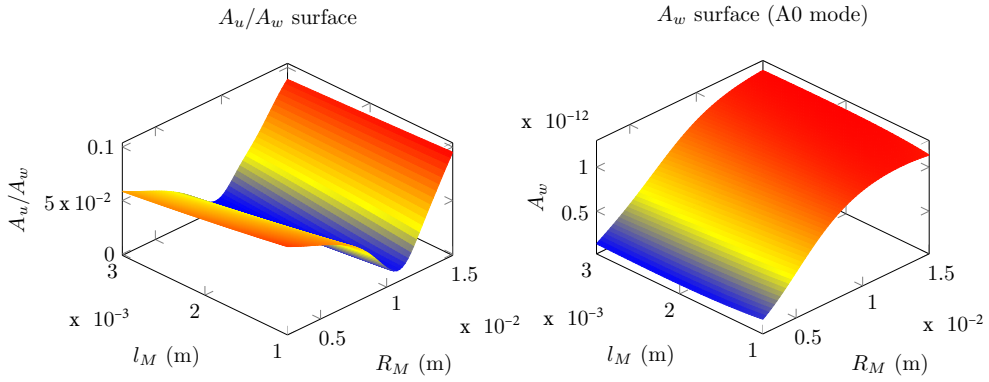


Figure 8. A_u/A_w and A_w surfaces predicted with the MLP with basic BP algorithm.

functions are predicted with the networks on a 60×60 grid of the design variables, as shown in Fig. 9. The values of $\frac{A_u}{A_w}$ and A_w are also scaled in the same way. From the figure, we can see that the multilayer networks trained with the LM algorithm approximate the surfaces better, compared to the networks trained with the basic BP algorithm.

Next we try the RBFNN with exact interpolation. All the 625 samples are used as the centers of the basis functions. The spread parameter is selected as 0.1 (normalized). In one run of the program, the time to build the network for the $\frac{A_u}{A_w}$ function is 0.457 s, and the time to build the network for the A_w function is 0.372 s. Figure 10 shows the $\frac{A_u}{A_w}$ and A_w surfaces predicted with the built networks, on a 60×60 grid of the design variables. From this figure, it could be observed that the performance of the RBFNN with exact interpolation is excellent. One notable phenomenon is that the noise in the surfaces relating to changing meshes are not big enough to cause overfitting. Compared to the multilayer feedforward networks, the additional advantage of the RBFNN is that the RBFNN model could be built very fast through simple algebraic calculations, instead of time-consuming iterative trainings.

Finally we test the RBFNN with nonexact interpolation. All the 625 samples are used as the centers of the basis functions. The spread parameter is selected as 0.18 (normalized). In one run of the program, the time to build the network for the $\frac{A_u}{A_w}$ function is 0.436 s, and the time to build the network for the A_w function is 0.447 s. Figure 11 shows the $\frac{A_u}{A_w}$ and A_w surfaces predicted with the networks, on a 60×60 grid of the design variables. From this figure, we can see that $\frac{A_u}{A_w}$ is somewhat smoother than it should be in the valley, with reference to Fig. 7. So the performance with nonexact interpolation is not as good as that with exact interpolation.

The built network models will be used in the optimizations of the EMAT.

6 Multiobjective optimizations of the EMAT accelerated with the neural-network metamodels

In this section we consider the multiobjective optimizations of the omnidirectional EMAT. The design variables are the lift-off of the magnet l_{M} and its radius R_{M}. The goal of optimization is to selectively generate the A0-mode Lamb waves. With the special structure of the EMAT, inevitably both the S0 mode and the A0 mode will be generated at the same time. So one objective function to minimize could be set as the ratio of the amplitude of the S0 mode to the amplitude of

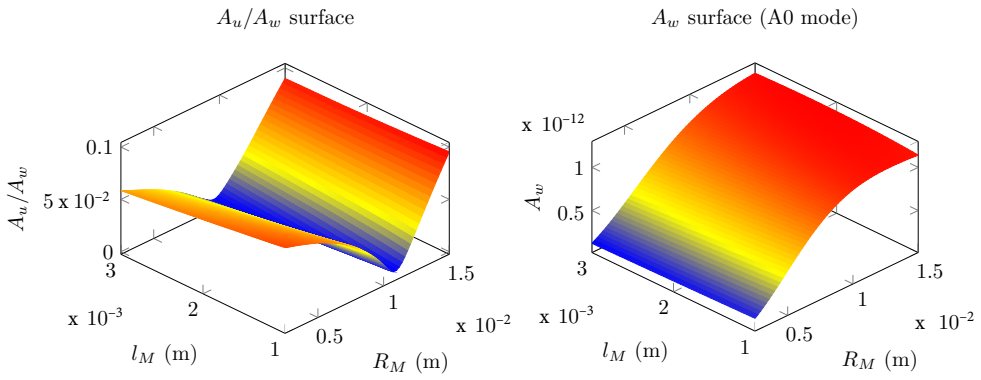


Figure 9. A_u/A_w and A_w surfaces predicted with the MLP with LM algorithm.

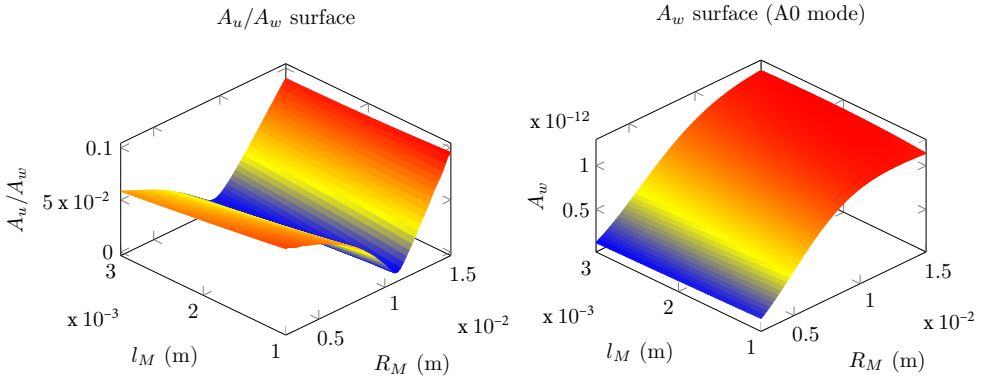


Figure 10. A_u/A_w and A_w surfaces predicted with the RBFNN with exact interpolation.

the A0 mode. Using the ratio as the objective function only tells one side of the story. We also want to maximize the A0 mode directly, or set the negative amplitude of the A0 mode as the other objective function to minimize. With these considerations, the problem of multiobjective optimization of the EMAT could be formulated as

$$\begin{cases} \text{minimize} \quad f_1(R_\mathrm{M}, l_\mathrm{M}) = \dfrac{A_u}{A_w} \\ \text{minimize} \quad f_2(R_\mathrm{M}, l_\mathrm{M}) = -A_w \end{cases}. \tag{54}$$

The design variables R_M and l_M have upper and lower bounds as $R_\mathrm{M} \in [2.5, 15]$ mm and $l_\mathrm{M} \in [1, 3]$ mm.

In multiobjective optimizations, it will be difficult to obtain one single solution, because the multiple objective functions often conflict with each other, so generally we could only obtain a set of solutions in which no solution is better than other solutions on every objective functions. This set of solutions are nondominated solutions, and they form a Pareto front in the criterion space where the first objective function is used as the x axis and the second objective function is used as the y axis, in the problem where there are two objective functions.

A popular method to solve the approximated Pareto front in the criterion space is the multiobjective genetic algorithm (MOGA; Deb, 2001). With the MOGA, the set of solutions on the Pareto front can be solved with one single run of the program. We have developed the MOGA program to implement the multiobjective optimizations of the EMAT discussed in this work. The implemented genetic operators include uniform mutation, nonuniform mutation, Gaussian mutation, whole arithmetic crossover, simple crossover and single arithmetic crossover. Real coding is employed. As described in Wang et al. (2017), the internal status of the MOGA program is tracked carefully to avoid unnecessary evaluations of the objective functions, so as to decrease the number of evaluations and the total time needed to accomplish the optimization. This is possible because the program is stochastic in nature, so when the value of an individual is not changed by the genetic operation, the corresponding objective function should not be computed.

The results of the multiobjective optimizations using the MOGA and the neural networks as the metamodels of the EMAT are shown in Fig. 12. The number of generations in the MOGA is 300, and the number of individuals in the population is 30. Total time of optimization and the number of evaluations of the objective functions for different networks and algorithms from the test runs of the MOGA program are summarized in Table 1. By comparison, if we optimize the

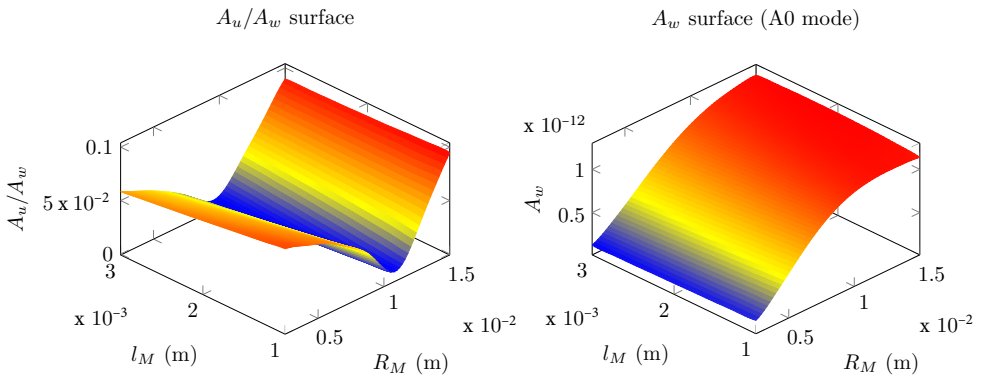


Figure 11. A_u/A_w and A_w surfaces predicted with the RBFNN with nonexact interpolation.

EMAT using the multifrequency approach without the proposed metamodels, nearly 9000 evaluations of the objective functions will cost around 20 days. One notable detail about Table 1 is that the number of evaluations of the objective functions is different for different networks and algorithms, even with the same number of generations and individuals in the genetic algorithm. This is understandable considering the mechanism we employed to reduce the number of evaluations of the objective functions. By tracking the internal status of the program and avoiding unnecessary evaluations, we incorporate additional randomness in the program, so the number of evaluations is not fixed.

The scatter plot of the values of the objective functions solved on the 25×25 discrete grid of the design variables is also drawn in Fig. 12 as comparison. Obviously the Pareto front should be approximately the tangent line at the lower left corner of the scatter plot. From the figure, it could be observed that the RBFNN with exact interpolation has the best performance. The multilayer network trained with the LM algorithm has a similar performance. The RBFNN with nonexact interpolation is acceptable, except near the leftmost of the Pareto front. The performance of the multilayer network trained with the basic back propagation algorithm is poor compared to the other networks, since the solved solutions already enter the region of the scatter points corresponding to the data on the discrete grid of the design variables. This is normal because with this network, the performance of approximation is poor with the selected training parameters.

Table 1. Total time of optimization and number of evaluations of the objective functions.

Network & algorithm	Time of optimization	Number of evaluations of the objective functions
Multilayer network with the BP algorithm	15.694 s	8937
Multilayer network with the LM algorithm	14.819 s	8929
RBFNN with exact interpolation	26.613 s	8940
RBFNN with nonexact interpolation	26.963 s	8952

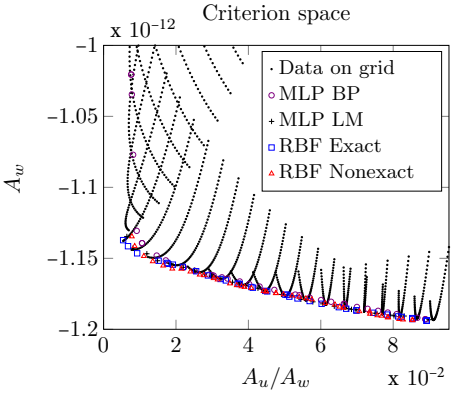


Figure 12. Pareto fronts solved with the MOGA and the neural-network metamodels and scatter plot of the objective function values solved on the discrete grid of the design variables.

7 Conclusions

In this work, we explored the artificial neural networks as metamodels of an EMAT model for the purpose of optimizations.

We are mostly interested in two kinds of neural networks: the multilayer feedforward networks (trained with the basic back propagation algorithm and the LM algorithm) and the radial basis function networks (with exact and nonexact interpolations). The basic structures and the training algorithms of these networks are reviewed, and the programs we have developed for this work are tested on an example problem.

Then the formulations of an axisymmetric EMAT were given, and the model of an omnidirectional EMAT for the generation of Lamb waves on a linearized steel plate was introduced; only the Lorentz force was considered. The axisymmetric model was divided into two geometries and three submodels. Radial and axial displacement components, corresponding to the S0-mode and the A0-mode Lamb waves respectively, of an observation point at the middle plane of

the plate were recorded to solve the two objective functions, one defined as the ratio of the amplitude of the S0 mode to that of the A0 mode, and the other defined as the negative amplitude of the A0 mode.

Next, three different approaches to calculate the amplitudes of the displacement components were discussed. The first one is solving the peaks of the envelopes of the time waveforms from time-domain simulations, which involves small time steps and is very time-consuming. The second one is transforming the input excitation signal into its spectrum and feeding the spectrum to the frequency model, then transforming the modified spectrum back to the time domain and solving the peaks of the envelopes of the time waveforms. The third one is using the magnitudes of the phasors of the displacement components directly, i.e., using a single frequency in the frequency model. The waveforms from the first and second approaches were compared. The reason why the third approach could be applied as an approximation when evaluating the amplitudes was also explained.

With the data on a discrete grid of the design variables as the reference, A_u/A_w and A_w surfaces were solved for the EMAT predicted with the neural networks. It could be observed that with the multilayer network trained with the basic BP algorithm, the A_u/A_w is smoother than it should be in the valley. With the multilayer network trained with the LM algorithm, the predicted surfaces were close to the reference surfaces. With the RBFNN with exact interpolation, the performance of approximation is good while with nonexact interpolation, the value of A_u/A_w is again smoother in the valley. For the multilayer networks, much work has to be done to tune the parameters of the network, i.e., the number of neurons in the hidden layer, and the number of iterations. The total time required to obtain the approximation with the RBFNN with exact interpolation is also much less than that with the multilayer networks, especially when the basic BP algorithm is applied.

Finally the obtained neural-network models, as approximations of the objective functions with the multifrequency approach, are applied in the multiobjective optimizations. The first objective is the ratio of the amplitude of the u waveform (S0 mode) to that of the w waveform (A0 mode), and the second objective is the negative amplitude of the w waveform. The multiobjective optimizations are accomplished with a MOGA program we have developed specifically for optimizations of EMATs. With the neural-network metamodels, we can use the more time-consuming multifrequency model, instead of using the approximate single-frequency model. From the results of the multiobjective optimizations, the performance of the RBFNN with exact interpolation is the best, next comes the multilayer network trained with the LM algorithm, then the RBFNN with nonexact interpolation. The performance of the multilayer network trained with the basic back propagation algorithm is the poorest.

Besides the work described above, the authors would like to suggest some topics to explore further, as potential future studies for interested readers. Some examples are as follows:

1. In this work implemented with our own code, the two objective functions are solved with their respective networks (two networks for two objective functions). What if we build all the neural networks with two outputs in the output layer directly?

2. In this work the parameters of the networks are mainly found by trial and error. Can we incorporate a systematic way to find the best parameters for the neural networks automatically?

3. Implementation of multiobjective optimizations of the EMATs with constraints. Anyway in applications, we will often meet with complex constraints, instead of the simple upper and lower bounds studied in this work.

All the listed topics demand modifications to the neural-network and/or the MOGA programs.

Data availability. No data sets were used in this article

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. This work was financially supported by the National Natural Science Foundation of China (grant no. 51677093), Tsinghua University Initiative Scientific Research Program (grant no. 20131089198), National Key Scientific Instrument and Equipment Development Project (grant no. 2013YQ140505), and China Scholarship Council (grant no. 201506215055).

Edited by: Andreas Schütze
Reviewed by: four anonymous referees

References

Auld, B. A.: Acoustic Fields and Waves in Solids, Krieger Publishing Company, 1990.

Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.

Dhayalan, R. and Balasubramaniam, K.: A hybrid finite element model for simulation of electromagnetic acoustic transducer (EMAT) based plate waves, NDT & E International, 43, 519–526, 2010.

Edwards, R., Sophian, A., Dixon, S., Tian, G., and Jian, X.: Dual EMAT and PEC non-contact probe: applications to defect testing, NDT & E International, 39, 45–52, 2006.

Gao, S., Dai, X., Liu, Z., and Tian, G.: High-Performance Wireless Piezoelectric Sensor Network for Distributed Structural

Health Monitoring, Int. J. Distrib. Sens. N., 12, 3846804, https://doi.org/10.1155/2016/3846804, 2016.

Hagan, M. T., Demuth, H. B., Beale, M. H., and Jesús, O. D.: Neural Network Design, 2nd Edn., 2014.

Hirao, M. and Ogi, H.: EMATs for Science and Industry: Noncontacting Ultrasonic Measurements, Springer, 2003.

Huthwaite, P., Ribichini, R., Cawley, P., and Lowe, M. J. S.: Mode Selection for Corrosion Detection in Pipes and Vessels via Guided Wave Tomography, IEEE T. Ultrason. Ferr., 60, 1165–1177, 2013.

Ida, N.: Engineering Electromagnetics, Springer, 2007.

Jafari-Shapoorabadi, R., Konrad, A., and Sinclair, A.: Improved finite element method for EMAT analysis and design, IEEE Transactions on Magnetics, 37, 2821–2823, 8th Joint Magnetism and Magnetic Materials International Magnetics Conference (MMM-INTERMAG), San Antonio, Texas, 7–11 January, 2001, 2001.

Konrad, A.: The numerical-solution of steady-state skin effect problems – an integrodifferential approach, IEEE T. Magn., 17, 1148–1152, 1981.

Ludwig, R. and Dai, X.: Numerical-simulation of electromagnetic acoustic transducer in the time domain, J. Appl. Phys., 69, 89–98, 1991.

Mirkhani, K., Chaggares, C., Masterson, C., Jastrzebski, M., Dusatko, T., Sinclair, A., Shapoorabadi, R., Konrad, A., and Papini, M.: Optimal design of EMAT transmitters, NDT & E International, 37, 181–193, 2004.

Preis, K.: A contribution to eddy-current calculations in plane and axisymmetric multiconductor systems, IEEE T. Magn., 19, 2397–2400, 1983.

Seher, M., Huthwaite, P., Lowe, M., Nagy, P., and Cawley, P.: Numerical Design Optimization of an EMAT for A0 Lamb Wave Generation in Steel Plates, in: 40th annual review of progress in quantitative nondestructive evaluation: incorporating the 10th international conference on barkhausen noise and micromagnetic testing, vols. 33A and 33B, edited by: Chimenti, D., Bond, L., and Thompson, D., vol. 1581 of AIP Conference Proceedings, Quantitat Nondestruct Evaluat Programs; Amer Soc Nondestruct Testing; NDE Centers, World Federat; Stresstech Grp & Int Conf Barkhausen Noise & Micromagnet Testing Org Comm; Int Comm NDT; Natl Sci Fdn Ind Univ Cooperat Res Ctr; Iowa State Univ, Ctr Nondestruct Evaluat, 340–347, 2014.

Seher, M., Huthwaite, P., Lowe, M. J. S., and Nagy, P. B.: Model-Based Design of Low Frequency Lamb Wave EMATs for Mode Selectivity, J. Nondestruct. Eval., 34, https://doi.org/10.1007/s10921-015-0296-6, 2015.

Sykulski, J.: New trends in optimization in electromagnetics, in: Institution of Engineering and Technology, 7th International Conference on Computation in Electromagnetics, CEM 2008, 44–49, 2008.

Thompson, R.: Model for electromagnetic generation and detection of Rayleigh and Lamb waves, IEEE T. Son. Ultrason., SU20, 340–346, 1973.

Thompson, R.: Generation of horizontally polarized shear-waves in ferromagnetic materials using magnetostrictively coupled meander-coil electromagnetic transducers, Appl. Phys. Lett., 34, 175–177, 1979.

Wang, S., Huang, S., Zhang, Y., and Zhao, W.: Multiphysics Modeling of a Lorentz Force-Based Meander Coil Electromagnetic Acoustic Transducer via Steady-State and Transient Analyses, IEEE Sens. J., 16, 6641–6651, 2016a.

Wang, S., Huang, S., Zhang, Y., and Zhao, W.: Modeling of an omni-directional electromagnetic acoustic transducer driven by the Lorentz force mechanism, VTT Symp., 25, 2016b.

Wang, S., Huang, S., Velichko, A., Wilcox, P., and Zhao, W.: A multi-objective structural optimization of an omnidirectional electromagnetic acoustic transducer, Ultrasonics, 81, 23–31, 2017.

Wilcox, P., Lowe, M., and Cawley, P.: The excitation and detection of Lamb waves with planar coil electromagnetic acoustic transducers, IEEE T. Ultrason. Ferr., 52, 2370–2383, 2005.