Advanced Numerical Methods Based on Artificial Intelligence

Levente Czumbil, Dan D. Micu, and Andrei Ceclan

Abstract The chapter presents some advanced numerical methods based on Artificial Intelligence (AI) techniques applied to specific electrical engineering problems. A theoretical description is done regarding the basic aspects of the nowadays most commonly used AI techniques: Neural Networks, Fuzzy Logic, and Genetic Algorithms respectively. The presented AI techniques are exemplified through two specific electrical engineering application implemented by the authors in their previous research projects. The first application consist in the identification of the optimal equivalent horizontally layered earth structure by means of a Genetic Algorithm according in site soil resistivity measurements. The second application provides a Neural Network alternative two evaluate the impedance matrix that describes the electromagnetic coupling between overhead powerlines and nearby underground pipelines for different separation distances and various vertically layered soil structures.

Keyword Artificial intelligence · Neural networks · Genetic algorithms · Electrical engineering application

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network

L. Czumbil · D. D. Micu (✉) · A. Ceclan
Numerical Methods Research Center, Department of Electrical Engineering, Technical University of Cluj-Napoca, Cluj-Napoca, Romania
e-mail: dan.micu@ethm.utcluj.ro

L. Czumbil
e-mail: levente.czumbil@ethm.utcluj.ro

A. Ceclan
e-mail: andrei.ceclan@ethm.utcluj.ro

N. Mahdavi Tabatabaei and N. Bizon (eds.), *Numerical Methods for Energy Applications*, Power Systems,
https://doi.org/10.1007/978-3-030-62191-9_4

EA Evolutionary Algorithms
FEM Finite Element Method
FL Fuzzy Logic
FLS Fuzzy Logic Systems
GA Genetic Algorithms
HVPL High Voltage Power Line
MGP Metallic Gas Pipeline
ML Measurement Location
NN Neural Networks
SM Surrogate Models

1 Introduction

The first definition for AI, which is still one of most accepted ones, was given in 1955 by McCarthy: "making a machine behave in ways that would be called intelligent if a human were so behaving" [1].

The AI technics represents a class of heuristic methods for solving the last decade's issues, that were born from the desire of implementing a system with the capacity to mimic the human mind. One of the most fundamental methods is the capacity of learning with or without external help and even with the purpose of permeant improvement. This method is usually used as a quick alternative for the old methods that requires a high effort and a long time of calculus compilation.

The main AI techniques that are used nowadays are Fuzzy Logic (FL), Neural Networks (NN), Genetic Algorithms (GA), Surrogate Models (SM) and Evolutionary Algorithms (EA) [2].

The main purpose of this chapter is to highlight the basic theoretical aspects of the most commonly used AI techniques (Genetic Algorithms, Fuzzy Logic and Neural Networks) and to exemplify how they could be implemented in case of specific electrical engineering applications.

Section 2 makes a brief introduction to the theoretical aspects regarding Genetic Algorithms based optimization techniques, presenting their structure, different chromosomal coding techniques and the basic GA operators. The next section describes how Fuzzy Logic Systems (FLS) work and main implementation steps (fuzzification, FL rule base interface and defuzzification respectively). Section 4 presents the basic theoretical aspects regarding Neural Networks: the structure of an artificial neuron, the main activation functions, the most commonly used NN architectures and training techniques.

The first demonstrative application (Sect. 5) shows how a genetic algorithm could be implemented to determine the optimal equivalent horizontal soil structure based on soil resistivity measurements. The second application (Sect. 6) exemplifies the implementation of Neural Networks in order to determine the inductive coupling

matrix in case of electromagnetic interference problems between overhead power line and nearby metallic pipelines.

2 Genetic Algorithms

Genetic Algorithms are part of the evolutionary computing strategies and represent a series of adaptive heuristic techniques based on the principle of natural selection: "The one who is best suited survives". The idea of evolutionary calculus was introduced in 1960 when several biologists began to use computers to simulate biological systems [3, 4].

Usually, genetic algorithms are used to solve multi-criteria optimization, planning or nonlinear search problems. They constitute a set of adaptive procedures that could find the solutions of a problem through a mechanism of natural selection and genetic recombination/evolution. The mechanism was introduced and analysed by J. Holland [5], being characterized by the fact that only the species (the solutions) that are better adapted to the environment (to the investigated problem) are able to survive and evolve over generations, while the less adapted ones will disappear. The likelihood of a species to survive and evolve over generations becomes greater as the degree of adaptation grows, which in terms of optimization it means that the solution is getting closer to an optimum.

2.1 Structure of a Genetic Algorithm

Genetic Algorithms start from an initial set of solutions, randomly generated or based on prior knowledge, referred as "population" in the literature. In this population, each individual represents a possible solution of the investigated problem and is defined by its "chromosome" structure (its coding). Within the GA the starting population is subjected to an iterative process, exemplified in Fig. 1, through which an optimal

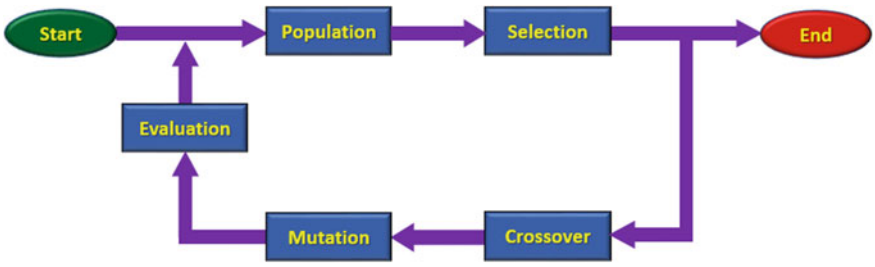

Fig. 1 The structure of a genetic algorithm

solution of the studied problem is determined. An iteration of this optimization/search process is known in literature as "a generation" of the genetic algorithm.

The iterative process that underlies any genetic algorithm can be defined by the following steps:

Step 1: Creation of a set of initial possible solutions ("individuals") that will form the starting population of the investigated problem;
Step 2: Selection based on an objective ("fitness") function of the individuals from the current generation population, that have adapted best to the needs of the problem that has to be solved;
Step 3: The selected individuals are subjected to genetic operators (such as "crossover" and "mutation") to form the population of the next generation;
Step 4: Evaluate the degree to which the members of the new generation correspond more adequately to the requirements of the studied problem;
Step 5: The population of old generation is abandoned, and the iterative process is resumed from Step 2.

Such a cycle is repeated until the best solution of the problem is identified or a predetermined number of generations/iterations has been reached [6, 7].

2.2 *Chromosome Structure of an Individual*

The chromosome structure of an individual defines how a candidate solution of the investigated problem is represented within a genetic algorithm. This constitutes the whole set of "genes", the parameters of an individual that must be determined/optimized for the studied problem. The genes of an individual can be represented either in binary form (Fig. 2a), through a finite string of 0 and 1 values, or in natural form (Fig. 2b) by a real value, generally in the range of 0 to 1.

In order to evaluate how each parameter ("gene"), that has to be optimized, correspond to the requirements of the investigated problem, a cost function, f_c, has to be defined, for each gene, g. The overall performance of an individual (possible solution) regarding the problem in question is determined by the GA objective (or "fitness") function, that is given by the weighted sum of these cost functions, see Eq. (1):

$$F = \frac{1}{n} \cdot \sum_{i=1}^{n} (p_i \cdot f_{Ci}(g_i)) \tag{1}$$

Fig. 2 Chromosome structure of an individual: **a** binary form, **b** natural form

where: n is the total number of parameters, p_i indicates the importance of the gene g_i, $i = 1..n$.

Within the iterative process of optimizing the solution, a minimization or maximization of the fitness function must to be achieved according to the investigated problem [8].

2.3 Selection Operator

The selection operator plays an important role in a genetic algorithm. This operator decides which of the individuals of a population will be able to participate in the formation of the next generation population. The purpose of the selection is to ensure more chances of "survival" / "reproduction" for the best performing individuals in a given population. The selection aims to maximize the performance of individuals (possible solutions to the problem in question).

2.4 Crossover Operator

The crossover operator is the most important operator in the optimization process. This operator applies to individuals in the current population for the purpose of generating individuals for the next generation, and thus ensuring the convergence of the problem. The mechanism of the crossover operator is highly dependent on the gene coding mode of the chromosome structure. Usually, the crossover operator applies to two parents (possible solutions) from the current population and provides two descendants (new solution configurations) for the next generation population. Descendants obtained through the crossover operation, will have characteristics from both parents. Due to its major importance, several crossing methods have been proposed in the literature [9, 10].

2.5 Mutation Operator

The mutation operator has the role of maintaining the diversity of the search space population by introducing individuals that could not be obtained through other mechanisms. This operator consists in randomly changing the value of a single gene/position in the chromosome structure of an individual. In the case of a binary gene coding, the process of mutation implies the negation of a bit in a gene, while in the case of natural form coding it implies a small variation of the value of a gene, see Eq. (2):

$$genA = genA + \xi \tag{2}$$

where: *genA* is the value of the parameter represented by gene *A*, and ξ is the applied perturbation [11].

Mutation is a probabilistic operator. Considering a population of N individuals, each one having n genes, the probability of a gene to undergo a mutation will be p_m, $m = 1..N\,n$. According to GA implementation these probability values could be equal or not equal for each gene.

There are several ways to apply a mutation operator. One of them would be the *change in formatting*. In this case for each position in the chromosome structure of an individual selected for mutation, a random number, q, is generated in the interval [0,1]. If $q > p_m$, then the mutation operator is executed, for that chromosome position, otherwise the position remains unchanged [10].

3 Fuzzy Logic

Fuzzy Logic (FL) offers an alternative, to classical linear equation-based methods, for dealing with problems that describe system operations. It is used especially when the connections between the input and output data of a system are too complex and cannot be defined exactly, due to a significant level of uncertainty in the analysed problem. In case of Fuzzy Logic Systems (FLS), conventional algorithms are replaced by a set of rules of the form *IF (premise) THEN (conclusion)*. This results in a heuristic algorithm that takes into account operator's experience in describing the investigated system.

The basis of the fuzzy set theory was laid by L.A. Zahed in 1965 [12]. From a mathematical point of view, the object of FL is to make a connection (application) between the set of input data of a system and its output values. This connection is made based on a set of *IF—THEN* type laws or reasoning. All the laws are evaluated in parallel and their order do not affect the outcome values.

Fuzzy Logic Systems work only with linguistic/fuzzy values. Therefore, all the input data must undergo a "fuzzification" process that transforms the actual values into fuzzy sets, and the obtained results has to be subjected to a "defuzzification" process for later use [13, 14], as in Fig. 3 can be seen.

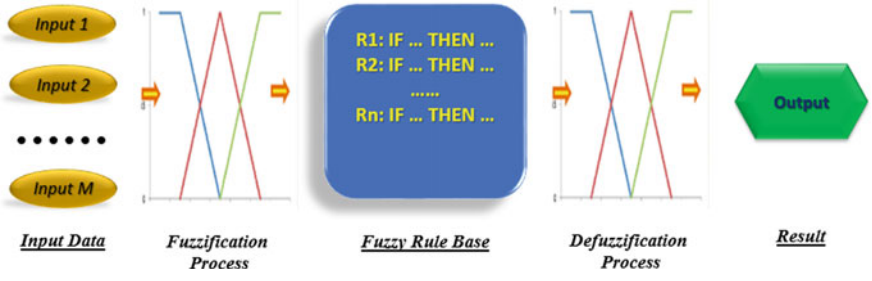

Fig. 3 Operation of a fuzzy logic system

3.1 Fuzzification

Fuzzification is the process by which the actual data provided at the input of a FLS block is transformed into linguistic variables defined by fuzzy sets. The notion of a fuzzy set has been introduced as a generalization of the concept of binary membership of an element to a set. Each fuzzy set is associated with a characteristic/membership function that provides a value in the [0,1] range. This value describes the degree of the belonging of an element to that fuzzy set [15], as in (3) is presented.

$$\mu_A : X \rightarrow [0, 1] \tag{3}$$

A fuzzy set is completely defined by its membership function. Most of the fuzzy sets used practical applications have a membership function defined over the set of real numbers. Therefore, is the most convenient way to express these membership functions as analytical equations [15].

3.2 Inference

The most important component in describing a fuzzy logic system is set of rules (laws) that are applied. The mathematical interpretation of these *IF – THEN* sentences is done through the *inference* process, which has several distinct parts. First, the premises are evaluated, which involves providing the input data and applying the fuzzy membership functions. Then the proper consequence of a fuzzy law is applied to the output values, this operation is known as an *implication*. The premise of a fuzzy rule can have several parts joined by fuzzy operators of "AND" or "OR" type [16], like in (4):

$$\begin{aligned} &\text{IF } x_1 \text{ is } A \text{ AND } x_2 \text{ is } B \text{ THEN } y \text{ is } V \\ &\text{IF } x_1 \text{ is } C \text{ AND } x_2 \text{ is } D \text{ THEN } y \text{ is } W \end{aligned} \tag{4}$$

where: x_1, x_2 are input values; A, B and C, D are fuzzy sets for input data x_1 and x_2 respectively; y is the FLS output and V, W are fuzzy sets corresponding to y.

Each part of the premise is evaluated separately, assigning a specific value to the fuzzy operators. The way in which these "AND" / "OR" operators are mathematically interpreted depends on the inference method adopted. The most commonly used inference methods in the literature are the Max–Min, the Max-Product and the Sum-Product respectively [15].

3.3 *Defuzzification*

The result of the inference are fuzzy sets attached to the FLS output values. In order to turn these fuzzy sets into real values, they must undergo a defuzzification process. The task of this operation is to determine that unique value from a given range belonging to each output data that best fits the resulting fuzzy sets.

Among the most common methods of defuzzification, in the literature, there are the centre of gravity method, the centre of sums method and the height method respectively [17].

4 Neural Networks

The most complex neural network in nature is the human brain, this inspired scientist to try to mimic it by designing Artificial Neural Networks (ANN). As in nature, ANN are constructed from smaller building blocks called neurons. The first attempt to schematically represent the mathematic model of an artificial neuron was made in the early 1940s by McCulloch and Pits [18].

As Fig. 4 shows, the architecture of an artificial neuron follows the structures of the biological neuron, being a system with variable number of m input data x_k, $k = 1..m$, and a single output value y. The m input values of an artificial neuron are multiplied by coefficients w_k, called the *weights*, and then summed together. The value thus obtained is added to a parameter b called *bias* value. The final sum, denoted by h, is applied as argument to the transfer function of the artificial neuron. This function is also known as activation function, f_a, in the literature and can have various mathematical implementations [19–21].

Thus, the output of an artificial neuron is generally described by Eq. (5):

$$y = f_a(h) \tag{5}$$

where:

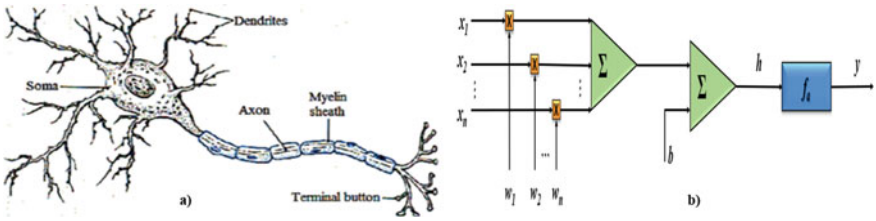

Fig. 4 Structure of a biological (**a**) and artificial (**b**) neurons [22]

$$h = \sum_{k=1}^{m} (x_k \cdot w_k) + b \tag{6}$$

The weights and the bias of an artificial neuron are adjustable parameters, and their values are determined during the neural network training process, in order to obtain the desired network behaviour. Therefore, when using a neuron, the output depends only on the set of input data and the used activation function.

4.1 Activation Functions

The activation function of a neuron is generally a bounded and monotony increasing function, as in Eq. (7), with values between 0 and 1 or between −1 and 1:

$$|f_a(h)| \leq M,\ M \in (0, +\infty), f_a'(h) > 0 \tag{7}$$

Each neuron of an artificial neural network can have its own activation function, however, usually, the same activation function is used for all neurons that form a layer. If back-propagation error technique is used to train the neural network, then it is necessary to know the first derivative of applied activation/transfer functions. In most application, for the output layer of neurons a linear transfer function is used while for the intermediate (hidden) layer neurons sigmoid type transfer functions are implemented.

4.2 Neuronal Networks Architecture

The output of a neural network is highly influenced by its architecture, how the neurons that form it are interconnected. As a NN architecture we understand the structure, more precisely the number of layers, the activation functions and the number of neurons used on each layer. A layer of neurons is formed by all the neurons that work in parallel with the same input data and have the same destination for their output data. Figure 5 shows the working configuration of a layer of neurons:

The weights of the neurons forming a layer can be grouped and placed in a matrix of weights W, while the bias values could be collected in a vector B [20], see Eq. (8):

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{1,n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{bmatrix} \tag{8}$$

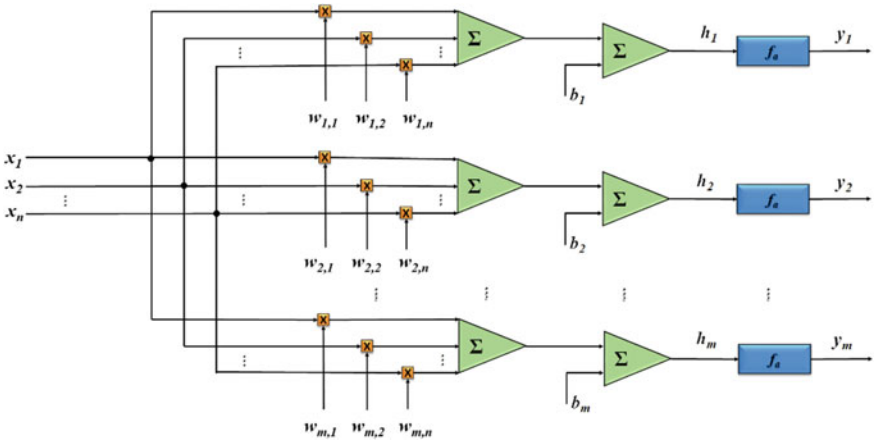

Fig. 5 Structure of a layer of neurons

Usually, the same activation function f_a is used for all the neurons from a specific layer. Therefore, the output values of a layer of neurons could be expressed as:

$$y_i = f_a(h_i),\ i = 1..n \tag{9}$$

with:

$$h_i = \sum_{k=1}^{m} \left(x_k \cdot w_{k,i}\right) + b_i,\ i = 1..n \tag{10}$$

The matrix form of the above equation in given by:

$$[H] = [W]^T \cdot [x] + [B] \tag{11}$$

The architecture of a specific NN could contain one or several layers of neurons. *Output layer* is called the layer of neurons that provides the final output data a neural network. This layer cannot be missing from the structure of any neural network. The neuron layers that interpose between the NN input data and the input values of the output layer are called *hidden layers*. In some literature references the first layer of neuros is called also as the *input layer* [21].

Figure 6 shows a simplified representation a *feed-forward neural network* with one hidden/input layer and one output layer. The following notations are used in Fig. 6: x_k, $k = 1..m$ for the m NN input data values; v_j, $j = 1..r$ for the r output values of the hidden/input layer neurons; y_i, $i = 1..n$ for the n output values of the NN; and $w^1{}_{k,j}$, respectively $w^2{}_{k,j}$ for the weights of the neurons in the two layers of the presented network configuration.

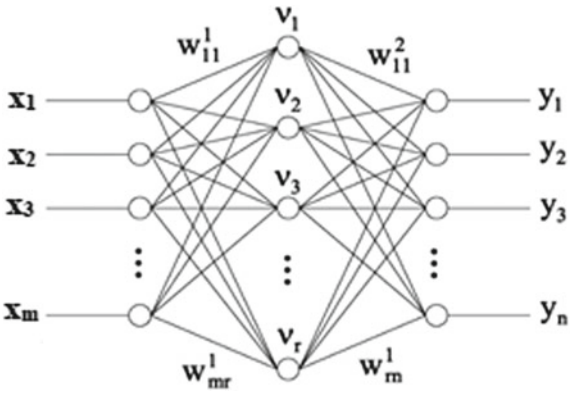

Fig. 6 Simplified diagram of a multi-layer feed-forward neural network [22]

Based on this simplified representation the mathematical form of the output data of a feed-forward neural network, with one hidden layer and one output layer, can be easily deduced according to equations (9–11). Thus, the arguments of the hidden layer neurons activation function are given by:

$$h_j^1 = \sum_{k=1}^{m} \left(x_k \cdot w_{k,j}^1 \right) + b_j^1, \;\; j = 1..r \tag{12}$$

while the output values of the hidden layer neurons are obtained through:

$$v_j = f_a^1\left(h_j\right) = f_a^1\left(\sum_{k=1}^{m} \left(x_k \cdot w_{k,j}^1 \right) + b_j^1 \right), \; j = 1..r \tag{13}$$

Therefore, the arguments of the output layer activation functions will be given by:

$$h_i^2 = \sum_{j=1}^{r} \left(v_j \cdot w_{j,i}^2 \right) + b_i^2 = \sum_{j=1}^{r} \left(f_a^1 \left(\sum_{k=1}^{m} \left(x_k \cdot w_{k,j}^1 \right) + b_j^1 \right) \cdot w_{j,i}^2 \right) + b_i^2, \;\; i = 1..n \tag{14}$$

Finally, the general NN output data could be evaluated with equation (15):

$$y_i = f_a^2\left(h_i^2\right) = f_a^2 \left(\sum_{j=1}^{r} \left(f_a^1 \left(\sum_{k=1}^{m} \left(x_k \cdot w_{k,j}^1 \right) + b_j^1 \right) \cdot w_{j,i}^2 \right) + b_i^2 \right), \;\; i = 1..n \tag{15}$$

Due to the above presented mathematical form, feed-forward neural networks can approximate/replace any kind of function. By using multiple hidden layers of neurons with sigmoid activation functions, a very good approximation could be obtained even for nonlinear relations between the NN input and output data. Linear

transfer functions applied on output layer neurons allows the network to provide any kind of output values. On the other hand, if it is desired to limit the output values, a sigmoid transfer function is advised to be used on the output layer [23].

A particular type of NN is the so-called *radial basis neural network*. This network contains a single hidden layer of neurons that uses the exponential function as transfer function. On the output layer, the linear activation function is used, similar to most feed-forward neural networks.

Recurrent neural networks have been also developed. In this case, the neurons from the hidden layers of the network, receive as input data and their own output value or the output data of nearby neuron layers. Recurrent neural networks are usually used in for the implementation of dynamic systems. For this reason, these networks are also called sequential networks [24].

However, the most common network architecture remains the feed-forward one due to the ease of implementation and training.

4.3 Training of Neural Networks

The process through which a neural network is taught to provide at its output the values of a specific desired function is called *training*. During the training process, the weights and bias values of the neurons are established so that the y outputs generated by network for a set of x input data, would be as close as possible to the target y^* values. Figure 7 graphically presents the error backpropagation principle in the NN training process.

Based to this principle the weights and bias values are continuously adjusted through an iterative process according to the error between the actual NN output values and the network desired ones. Several training algorithms were developed from this basic error backpropagation principle like: gradient descent algorithms; conjugate gradient algorithms; quasi-Newton algorithms; and Jacobian based Levenberg–Marquardt and Bayesian Regularization algorithms [24].

Usually, a large number of input/target output pairs (x, y^*) are used to train a neural network. The values of the weights and biases depend on the applied training algorithm and error evaluation technique. The evaluation of the error between the provided NN output data and the target values is done through a *cost* ("fitness") *function*. Since

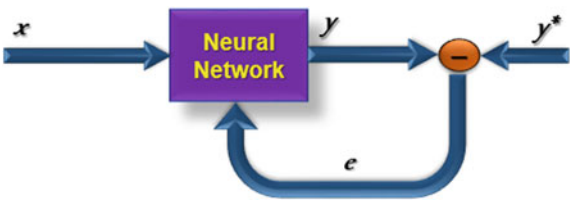

Fig. 7 Error backpropagation principle used to train neural networks

the cost functions express the deviation from the desired NN behaviour, these functions are also called as *quality indicators* of the networks. The most commonly used quality indicator is the mean square error [24], see Eq. (16).

$$MSE = \frac{1}{n} \cdot \left(\sum_{i=1}^{n} \left(y_i^* - y_i \right)^2 \right) \tag{16}$$

5 Identification of the Proper Equivalent Multi-layer Earth Structure Through a Genetic Algorithm Based AI Technique

5.1 Description of the Presented Application

Several electrical engineering applications like grounding grid design for power substations or strategic buildings, cathodic protection design of underground metallic gas or oil pipelines, design of lightning protection system require as a first step a proper knowledge of the earth structure from an electrical (soil resistivity) point of view. The following application meant to exemplify how genetic algorithm based optimization techniques could be applied in electrical engineering.

In order to determine the equivalent multi-layer earth structure corresponding to on site soil resistivity measurements the authors have developed an AI based optimization technique [25, 26]. The implemented genetic algorithm identifies the optimum value of the resistivity and the width of each soil layer considering horizontal multi-layer earth model, in order to reconstruct the measured apparent soil resistivity data.

The developed GA optimization will be applied to determine the proper multi-layer soil structure for two different locations (ML1 and ML2) where on-site Wenner type soil resistivity measurements were carried out. Obtained earth structure data are compared to soil configurations provided by dedicated software applications (the RESAP tool from the CDEGS software package [27]).

5.2 Implemented Genetic Algorithm

The implemented GA starts from a population of 30 individuals, randomly generated, each of them representing a possible configuration of the multi-layered soil model that has to be determined. The chromosome structure of each possible solution contains the resistivity, respectively the thickness of the equivalent soil layers.

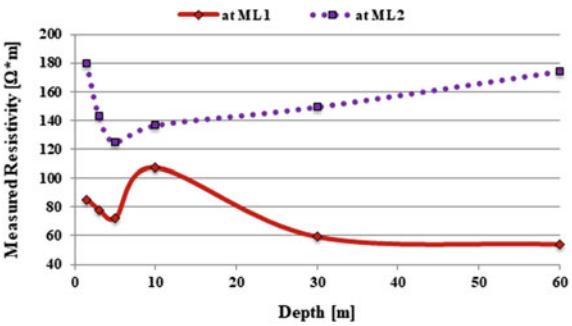

Fig. 8 Measured Wenner apparent soil resistivity curve at location MP1 and MP2

To determine the optimal soil structure, the implemented GA uses a cost function, Eq. (17), that evaluates the mean square error between the Wenner apparent resistivity curve, obtained through soil resistivity measurements (see Fig. 8), and the apparent resistivity curve, related to a possible multi-layer earth configuration:

$$MSQ_{Err} = \frac{1}{n}\sum_{i=1}^{n}[\rho_a(d_i) - \rho_{Ea}(d_i)]^2 \tag{17}$$

where: MSQ_{Err} is the mean square error; n is the number of measurement points; $\rho_a(d_i)$ is the apparent soil resistivity value measured through the Wenner method; and $\rho_{Ea}(d_i)$ is the apparent soil resistivity value corresponding to a horizontal equivalent multi-layered earth model, numerically computed for a depth d_i, $i = 1..N$.

For the numerical evaluation of the apparent soil resistivity, related to a possible soil configuration, the following equation was adopted [28]:

$$\rho_{Ea}(d) = \rho_1 \cdot [1 + 2 \cdot F_L(d) - F_L(2 \cdot d)] \tag{18}$$

where the value of the $F_L(d)$ function is given by the semi-infinite integral:

$$F_L(d) = 2 \cdot d \cdot \int_0^{\infty} \frac{K_{L1} \cdot e^{-2\cdot\lambda\cdot h_1}}{1 - K_{L1} \cdot e^{-2\cdot\lambda\cdot h_1}} \cdot J_o(\lambda \cdot d) \cdot d\lambda \tag{19}$$

with $J_0(\lambda \bullet d)$ the first kind, zero order, Bessel function and K_{L1} a coefficient determined by:

$$K_{L1} = \frac{k_1 + K_{L2} \cdot e^{-2\cdot\lambda\cdot h_2}}{1 - k_1 \cdot K_{L2} \cdot e^{-2\cdot\lambda\cdot h_2}} \tag{20}$$

and

$$K_{Lj} = \frac{k_j + K_{Lj+1} \cdot e^{-2\cdot\lambda\cdot h_{j+1}}}{1 - k_j K_{Lj+1} \cdot e^{-2\cdot\lambda\cdot h_{j+1}}}, \quad j = 1..L-2 \tag{21}$$

where $K_{LL\text{-}1} = k_{L\text{-}1}$, L being the number of horizontal layers, h_j the thickness of the jth layer and k_j the reflection coefficient between layers j and $j + 1$ with soil resistivity ρ_j and ρ_{j+1} respectively:

$$k_j = \frac{\rho_{j+1} - \rho_j}{\rho_{j+1} + \rho_j} \tag{22}$$

5.2.1 The Iterative Optimization Process

To obtain the optimal equivalent earth horizontal model, the set of possible solutions, from the initial GA population, is involved in an iterative process defined by the following steps [25]:

Step 1: The cost function is evaluated for all the possible soil configuration from the current GA population and the best suited ones are directly transferred to the next GA generation;
Step 2: Two soil configuration are randomly selected and subjected to the crossover operator to obtain two new equivalent soil configurations with lower cost function values for the GA next generation;
Step 3: The previous step is repeated until the next GA generation will have the same number of individuals (possible solutions) as the current one;
Step 4: In order to maintain solution diversity four soil configurations are randomly selected and the mutation operator is applied on them;
Step 5: The iterative GA optimization process restarted form *Step 1*.

The maximum number of GA iterations was set to $N = 2000$, a value identified by the authors to be high enough to obtain accurate soil configurations. This way, the implemented GA identifies the optimal parameters of a specific multi-layer earth structure, according to on site Wenner apparent soil resistivity measurements.

5.2.2 Chromosome Structure

Each possible soil configuration solution is represented in the GA optimization process by its chromosome structure formed by the resistivity and thickness of each soil layer scaled to [0,1] range, as in Eq. (23).

$$C = \{\rho_1, h_1, \rho_2, h_2, \ldots, \rho_L\} \tag{23}$$

5.2.3 Crossover Operator

During the crossover process, six new soil configurations are obtained from the initial two solutions selected for crossover recombination, applying three different crossover operators. The first two configuration (GA children) are obtained through an arithmetic crossover operator, Eq. (24):

$$\begin{aligned} C_1^t &= \alpha \cdot P_1^t + (1-\alpha) \cdot P_2^t \\ C_2^t &= \alpha \cdot P_2^t + (1-\alpha) \cdot P_1^t \end{aligned} \tag{24}$$

where: α is a randomly determined scaling factor, t denotes the tth parameter of an equivalent soil model, C_i and P_j represent the ith GA child configuration and jth GA parent soil configuration.

Another two new soil configurations are obtained using a Max–Min type crossover operator, Eq. (25):

$$\begin{aligned} C_3^t &= \min(P_1^t, P_2^t) \\ C_4^t &= \max(P_1^t, P_2^t) \end{aligned} \tag{25}$$

The last two GA child soil configurations are generated applying the classical cut-point crossover operator [8], Eq. (26):

$$\begin{aligned} C_5 &= \left(P_1^1 \cdots P_1^k \; P_2^{k+1} \cdots P_2^r\right) \\ C_6 &= \left(P_2^1 \cdots P_2^k \; P_1^{k+1} \cdots P_1^r\right) \end{aligned} \tag{26}$$

where k is a randomly selected cut point and r is the total number of chromosome structure parameters.

From these six GA child configurations the best two ones with lower cost function values are transferred the next GA generation population.

5.2.4 Mutation Operator

Within the mutation process, each parameter that has to be optimized from a possible multi-layer earth configuration is subjected to a probabilistic test. If the test is passed, then the value of the selected parameter is slightly changed through the following arithmetic mutation operator:

$$C^t = C^t + (0.5 - \alpha) \cdot M \tag{27}$$

with α a random value from the [0,1] range and M a predefined mutation coefficient.

5.3 Computed Equivalent Soil Models

The above presented GA optimization process was applied by the authors to determine the equivalent soil structure based on the on-site Wenner soil resistivity measured at location ML1 and ML2 (see measured apparent soil resistivity curves from Fig. 8) considering a three horizontal layer earth structure. To validate the obtained multi-layer earth configurations a comparison has been done with the RESAP module of CDEGS software package (see Table 1).

Based on the layer resistivity and thickness values obtained through the implemented GA optimization process and the RESAP module respectively (see Table 1, Fig. 9a and Fig. 10a) the apparent soil resistivity curves were generated according

Table. 1 Obtained equivalent three horizontal layer soil models

		$\rho_1[\Omega/m]$	$h_1[m]$	$\rho_2[\Omega/m]$	$h_2[m]$	$\rho_3[\Omega/m]$	$h_3[m]$
ML1	CDGES	80.77	0.99	82.52	13.61	49.31	Inf
	GA	83.84	5.25	101.18	4.63	51.78	Inf
ML2	CDGES	210.69	1.29	109.02	5.16	168.55	Inf
	GA	178.678	1.29	112.34	6.67	174.99	Inf

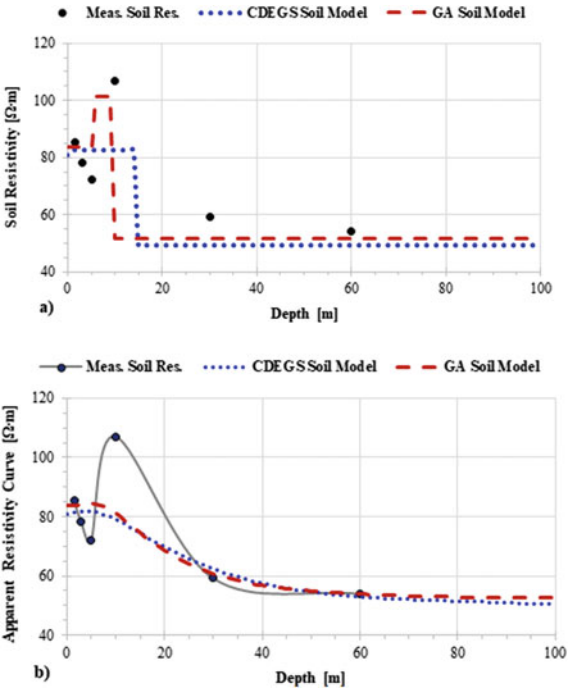

Fig. 9 Obtained three-layer equivalent earth structure (**a**) and Apparent soil resistivity curves (**b**) with the CDEGS software and the implemented GA for measurement location ML1

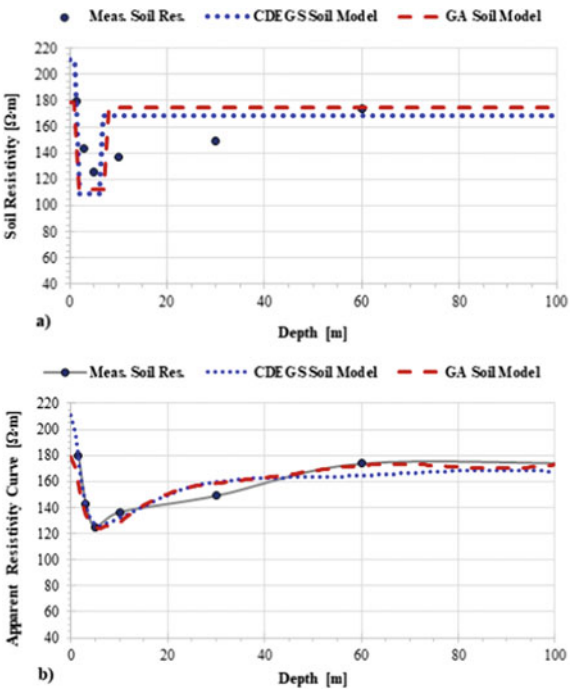

Fig. 10 Obtained three-layer equivalent earth structure (**a**) and Apparent soil resistivity curves (**b**) with the CDEGS software and the implemented GA for measurement location ML2

to equations (18–22) and compared to the on-site Wenner apparent soil resistivity measurements (see Fig. 9b and Fig. 10b).

For measurement location ML1, the average deviation from the measured apparent soil resistivity curve is 7.66% for the equivalent three-layer soil structure provided by the RESAP module of the CDEGS software package, while the average deviation for the soil structure provided by the implemented genetic algorithm is 7.16% (see Fig. 9b).

In case of measurement location ML2 the average deviation from the measured apparent soil resistivity curve are 3.83% for RESAP and 3.30% with the implemented GA optimization process (see Fig. 10b).

Similar comparisons have been carried out by the authors for uniform and two-layer horizontal earth structures in [25] and [26]. Based on the obtained results it can be concluded that the implemented GA provides an accurate alternative to evaluate the equivalent multi-layer earth structure using to on-site apparent soil resistivity measurements.

The above presented multi-layer soil structure GA optimization technique was also applied by the authors at archaeological sites in order to identify and establish the trajectory of buried walls, according to the obtained equivalent earth configurations [29].

6 Neural Network Implementation to Evaluate the Inductive Coupling Matrix in Case of a HVPL – MGP Electromagnetic Interference Problem

6.1 Description of the Studied Problem

Due to economic policies meat to limit construction costs and to environmental regulations meant to protect wildlife and nature, the access of utility systems to new right-of-ways is highly limited. Therefore, in many situations gas, oil or water transportation metallic pipelines are forced to share the same distribution corridor with high voltage power lines and/or AC electrical railway systems (see Fig. 11) and to be exposed to induced AC currents and voltages [30, 31].

In case of underground or above ground metallic pipelines, the induced electromagnetic interferences produced by nearby high voltage power lines could be dangerous on both the operating personnel (that may be exposed to electric shocks), and to the structural integrity of the pipeline, due to corrosion phenomena [31].

Induced AC currents and voltages may appear as effect of inductive, conductive or capacitive coupling mechanisms. However, during power line normal operating conditions, only the inductive coupling, described by the self and mutual inductance matric, has to be considered for underground pipelines. Conductive and capacitive coupling may be, also, neglected when a phase to ground fault happens on the power line far away from the common distribution corridor [30, 32].

To evaluate the self and mutual inductance between all the present conductors in the analysed problem geometry (phase wires, sky wires and pipelines) the magnetic vector potential must be evaluated on the cross section of these conductors as presented in [33, 34]. The longitudinal z-direction component of the magnetic

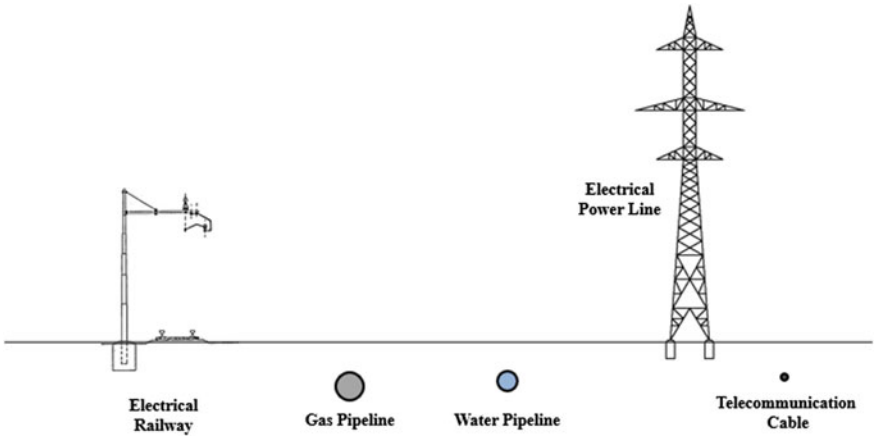

Fig. 11 Common distribution corridor of multiple utilities

vector potential A_z and the total current density J_z are described by the following system of differential equations:

$$\begin{cases} \dfrac{1}{\mu_0 \mu_r} \cdot \left[\dfrac{\partial^2 A_z}{\partial x^2} + \dfrac{\partial^2 A_z}{\partial y^2} \right] - j\omega\sigma A_z + J_{sz} = 0 \\ -j\omega\sigma A_z + J_{sz} = J_z \\ \displaystyle\iint_{S_i} J_z ds = I_i \end{cases} \tag{28}$$

where σ is the conductivity, ω is the angular frequency, μ_0 is the magnetic permeability of free space ($\mu_0 = 4 \cdot \pi \cdot 10^{-7}$ H/m), μ_r is the relative permeability of the environment, J_{sz} is the source current density in the z-direction and I_i is the imposed current on conductor i of S_i cross section.

To solve this differential equation system, the finite element method (FEM) is recommended to be used. Although the calculation process based on FEM, used in the hybrid method presented in [34], provides accurate solution for the magnetic vector potential, regardless of the complexity of the problem, the computation time of the method increases with the complexity of the geometry, the size of the discretization network, the characteristics of the material and the number of parameters being evaluated.

Therefore, the authors have implemented a neural network solution to evaluate the inductive coupling matrix for a specific electromagnetic interference problem between a 220 kV/50 Hz overhead High Voltage Power Lines (HVPL) and underground Metallic Gas Pipeline (MGP) [35], considering a stratified soil structure for the common distribution corridor with three vertical layers (see Fig. 12).

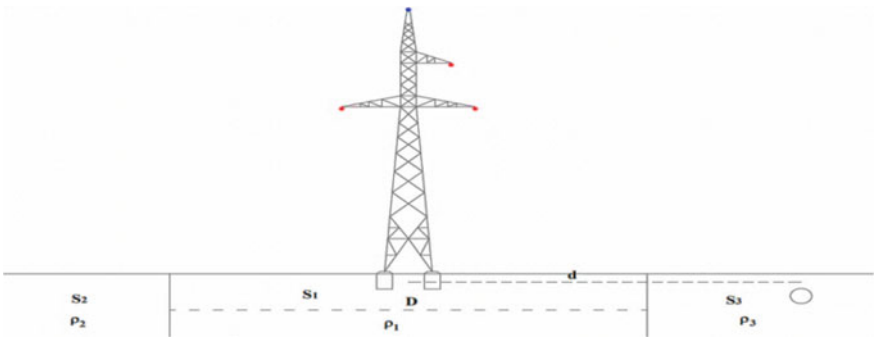

Fig. 12 Interference problem HVPL-MGP with vertically layered earth

6.2 Proposed Neural Network Solution

Once the proposed neural network solution will be trained it will have to be able to instantly evaluate the self and mutual inductance matrix for any possible geometric configuration of the investigated electromagnetic interference problem. Therefore, to implement the proposed NN, the input and desired output data values must be analysed. The following geometrical and electrical parameters of the studied problem were chosen as input values:

- d—HVPL-MGP separation distance (with variation in the 0–1000 m range);
- ρ_1—middle layer resistivity S1 (with variation in the 10–5000 Ω m range);
- D—middle layer width S1 (with variation in the 20–1200 m range);
- ρ_2, ρ_3 (considering $\rho_2 = \rho_3$) —sideways layers resistivity S2 and S3 (with variation in the 10–5000 Ω m range);

Tacking into the account that the inductance matrix is a symmetrical one, the proposed NN should provide only the elements above the main diagonal. For the investigated HVPL-MGP interference problem (three phase wires, one sky wire and one underground pipeline) these inductance elements are: $L_{11}, L_{12}, L_{13}, L_{14}, L_{15}, L_{22}, L_{23}, L_{24}, L_{25}, L_{33}, L_{34}, L_{35}, L_{44}, L_{45}, L_{55}$, with L_{ii} representing the self-inductance of conductor i ($i = 1.0.3$ for phase wires, $i = 4$ for the sky wire and $i = 5$ for the underground pipeline) and L_{ij} representing the mutual inductance between conductor i and j.

Due to the large variation range of the inductance matrix elements value (the self-inductance values are much higher than the mutual inductance values), it was concluded to implement three different neural networks: NN1 for the self-inductance values ($L_{11}, L_{22}, L_{33}, L_{44}, L_{55}$), NN2 for the MGP mutual inductances ($L_{15}, L_{25}, L_{35}, L_{45}$) and NN3 for the remaining mutual inductances between HVPL conductors. This way the complexity of the implemented NN will be reduced, so that the required training time will also be reduced, and the obtained results accuracy will be increased.

6.3 Matlab Implementation of Proposed Neural Network

The Neural Networks toolbox from the MATLAB software package [24] was used to implement, test and validate the proposed NN solution. A feed-forward architecture with two hidden layers and an output layer was chosen (as in Fig. 13).

To identify the optimal configuration of the chosen NN architecture different transfer functions and various number of neurons on the NN hidden layers were tested. The number of neurons on each hidden layer was varied between 5 and 30 with a step of 5. The “tansig” (sigmoid tangent) and “logsig” (logarithmic sigmoid) transfer functions were tested for the NN hidden layer neurons while the “purelin” (linear) transfer function was used for the output layer neurons. To automatically generate and test all these different possible NN configurations a Matlab code “.m”

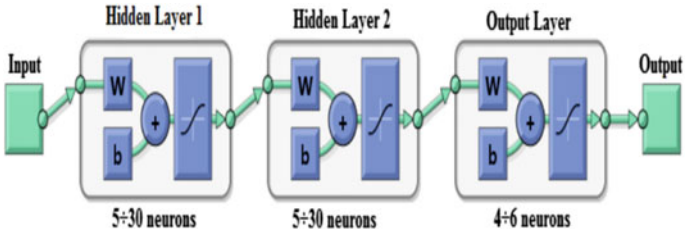

Fig. 13 Implemented feed-forward architecture with two hidden layers and an output layer

file was implemented, using the *feedforwardnet* Matlab function [24]:

$$\text{net} = \textit{feedforwardnet}(\text{hiddenSizes}, \text{trainFcn}) \tag{29}$$

where: *net* is the created feed-forward neural network, *hddinSizes* is a vector of values specifying the number of neurons used on each hidden layer and *trainFcn* is a vector of strings defining the transfer function used on each NN layer.

To create a useful training database for the investigated HVPL-MGP electromagnetic interference problem approximately 4000 inductance matrixes were determined through FEM analysis for various problem geometries. The HVPL-MGP separation distance was varied between 0 and 1000 m, the resistivities of the vertical soil layers were varied between 10 Ω m and 5000 Ω m while the width of the middle soil layer was varied between 20 and 1200 m. Table 2 shows some of the HVPL-MGP problem geometries used to train the proposed neural networks. different configurations used to stimulate the NN.

The NN training process took between 1 and 25 min depending on the NN configuration complexity. The Levenberg–Marquardt training method ("trainlm") was used with a mean square error ("mse") cost function on a i7-3632QM 2.2 GHz Intel Core

Table. 2 Different problem geometry configurations used for NN training

Case no	d $[m]$	D $[m]$	ρ_1 $[\Omega \cdot m]$	ρ_2 $[\Omega \cdot m]$	ρ_3 $[\Omega \cdot m]$	**Case No**	d $[m]$	D $[m]$	ρ_1 $[\Omega \cdot m]$	ρ_2 $[\Omega \cdot m]$	ρ_3 $[\Omega \cdot m]$
8	5	60	500	50	500	2134	0	550	50	250	50
104	100	60	150	250	150	2301	20	550	30	250	30
206	20	60	50	500	50	2532	100	550	100	500	100
373	100	60	500	750	500	2751	500	550	30	100	30
481	150	60	500	250	500	2914	5	1050	10	250	10
692	1000	60	750	50	750	3096	20	1050	100	250	100
875	20	120	750	100	750	3274	100	1050	500	1000	500
1064	50	120	750	1000	750	3545	750	1050	30	750	30
1231	500	120	100	30	100	3754	5	1500	50	30	50

Table. 3 HVTL-MGP problem geometries used for the NN testing procedure

Case no	d $[m]$	D $[m]$	ρ_1 $[\Omega \cdot m]$	ρ_2 $[\Omega \cdot m]$	ρ_3 $[\Omega \cdot m]$	Case No	d $[m]$	D $[m]$	ρ_1 $[\Omega \cdot m]$	ρ_2 $[\Omega \cdot m]$	ρ_3 $[\Omega \cdot m]$
1	310	800	900	850	900	85	310	800	900	850	900
13	105	1100	550	550	550	97	170	700	300	350	300
25	250	800	150	150	150	109	240	500	80	750	80
37	340	400	600	150	600	121	420	100	550	20	550
49	170	800	650	750	650	135	105	1200	250	950	250
54	55	1000	900	400	900	148	85	400	140	160	140
61	40	200	600	800	600	176	15	300	140	700	140
73	120	900	750	350	750	198	10	1000	200	750	200

PC, with a 64-bit operating system and 8 GB RAM memory. To train the implemented NN configurations the *train* Matlab function was applied [24].

6.4 Obtained NN Results

In order to determine the accuracy of the generated NN architectures and to identify the optimal NN configuration for each of the three implemented NN solutions (NN1, NN2 and NN3 respectively) an addition set of approximatively 200 randomly generated, testing HVPL-MGP problem geometries were used. These testing HVPL-MGP problem geometries were not supplied to the implemented NN configuration during the NN training process. Table 3 shows some of the testing HVPL-MGP problem geometries.

To identify the optimal NN configurations the evaluation error of the provided NN output data was analysed for both the training and testing data sets [35, 36]. To obtain NN provided output data for the training and testing HVPL-MGP problem geometries the *sim* Matlab function was applied.

For the neural network meat to evaluate the self-inductance values of the conductors (NN1 network) the best identified NN configuration was a feed-forward architecture with 15 neurons on the first hidden layer and 25 neurons on the second hidden layers, with "tansig" transfer function on both hidden layers. The obtained average evaluation errors are 0.064% for the testing geometries and 0.043% for the training geometries. The maximum recorded evaluation error was 0.77%. Figure 14 presents the evaluation error distribution on different error classes for both training and testing HVPL-MGP problem geometries.

In case of the neural network implemented to compute the mutual inductance values that define the electromagnetic coupling between MGP and the nearby HVPL (NN2 network) the best NN configuration has 30 neurons on the first hidden layer and 20 neurons on the second layer with "logsig" transfer function. The average

Fig. 14 Evaluation error distribution for the optimal NN1 architecture

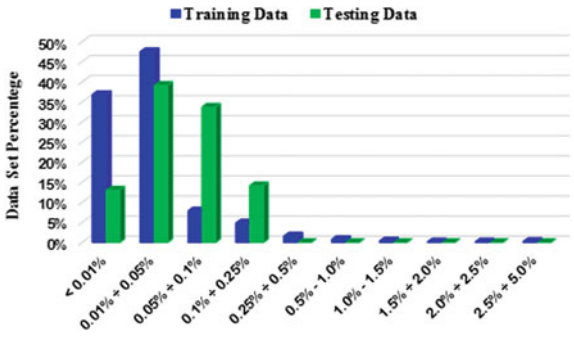

Fig. 15 Evaluation error distribution for the optimal NN2 architecture

evaluation error was around 0.060% for both testing and training data sets, while the maximum recorded evaluation error was 2.67%. The evaluation error distribution over the analysed error classes for the testing and training HVPL-MGP problem geometries is presented in Fig. 15.

For the neural network used to compute the mutual inductance values between HVPL conductors (NN3 network) the best NN configuration has 25 neurons, respectively 15 neurons with "tansig" transfer function on the NN hidden layers. The maximum recorded evaluation error is 2.56% while the average evaluation error is around 0.030% for both testing and training data sets. Figure 16 shows the obtained evaluation error distribution over different error classes:

The implemented NN configurations allow to evaluate the inductance matrix values for any HVPL-MGP problem geometry. Table 4 shows the self and mutual inductance values obtained for a HVPL-MGP problem geometry with a 30 m separation distance between HVPL and MGP; with $\rho_1 = 30\ \Omega$ m, $\rho_2 = \rho_3 = 500\ \Omega$ m and a 20 m width for the middle earth layer. Using the self and mutual inductance values provided by the implemented neural network configurations the equivalent electrical circuit of the investigated HVPL-MGP electromagnetic interference problem could be constructed according to [34, 37].

The *InterfStud* software application developed by the authors [38] automatically creates the above-mentioned equivalent circuit model and evaluates the induced AC

Fig. 16 Percentage error distribution for the optimal NN3 network

Table. 4 Obtained inductive coupling matrix through NN implementation

Self and mutual inductances [μH/m]					
	PhW A	PhW B	PhW C	SkyW	Pipe
PhW A	2.45	1.234	1.110	1.187	0.82
PhW B	1.234	2.45	1.100	1.073	0.84
PhW C	1.110	1.100	2.45	1.073	0.80
SkyW	1.187	1.073	1.073	8.74	0.79
Pipe	0.822	0.842	0.80	0.795	2.28

currents and voltages in the MGP. Figure 17 presents the obtained induced AC voltages for the three different problem geometries [35], considering a 10 km long parallel HVPL-MGP exposure, a 130 MVA power load on HVPL with a 0.94 power factor (a 350 A symmetrical current load):

- ***Geom* 01**: A 30 m separation distance, with soil structure: $\rho_1 = 30\ \Omega$ m, $\rho_2 = \rho_3 = 500\ \Omega$ m, and 20 m middle layer width;
- ***Geom* 02**: A 50 m separation distance, with soil structure: $\rho_1 = 10\ \Omega$ m, $\rho_2 = 100\ \Omega$ m, $\rho_3 = 500\ \Omega$ m, and 30 m middle layer width;

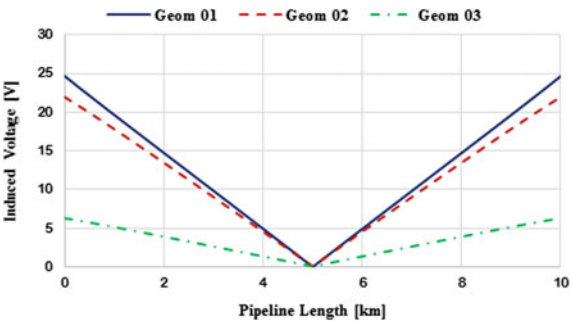

Fig. 17 Induced voltage in MGP for different HVPL-MGP problem geometries

- ***Geom* 03**: A 150 m separation distance, with soil structure: $\rho_1 = \rho_2 = 100\ \Omega$ m $\rho_3 = 1000\ \Omega$ m, and 100 m middle layer width.

7 Conclusions

This chapter starts with a brief introduction to artificial intelligence (AI) based advanced numerical methods applied in engineering, making a summary of the most commonly used AI techniques (Genetic Algorithms, Fuzzy Logic and Neural Networks, Sects. 2–4) and new approaches in the field (through two demonstrative applications).

The first application (Sect. 5) presents a genetic algorithm implementation to determine the equivalent horizontal soil structure based on Wenner on-site soil resistivity measurements. A proper knowledge of the earth structure is required in electrical engineering application like grounding grid design for power substations, cathodic protection design of underground metallic gas or oil pipelines, design of lightning protection.

The presented multi-layer soil structure GA optimization technique was also applied by the authors at archaeological sites in order to identify and establish the trajectory of buried walls, according to the obtained equivalent earth configurations.

In the second presented application (Sect. 5) a neural network based artificial intelligence technique has been implemented to evaluate the inductive coupling matrix of a specific HVPL-MGP electromagnetic interference problem. The proposed neural network approach reduces considerably the required computation time. From Figs. 14–16 it can be observed that the evaluation error produced by the identified optimal NN architecture are usually less than 0.1% in comparison to the finite element results considered as reference. Therefore, the implemented neural network solution to evaluate the self and mutual inductance values is a very effective one, especially if we take into account the fact that the solutions provided by neural networks are obtained almost instantaneously and can be used to evaluate the induced currents and voltages.

References

1. McCarthy J, Minsky ML, Rochester N, Shannon CE (1995) A proposal for the dartmouth summer research project on artificial intelligence. https://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html. Accessed 31 Aug 1995
2. Brunette ES, Flemmer RC, Flemmer CL (2009s) A review of artificial intelligence. 4th International conference on autonomous robots and agents (ICARA), pp 385–392, Wellington, New Leeland, February 10–12 2009
3. Rechenberg I (1971) Evolutionsstrategie: Optimierung technischer Systeme nach PrinLipien der biologischen evolution", Dr. Ing. thesis, Technical University of Berlin, Department of Process Engineering, Berlin

4. Beyer HG, Schwefel HP (2002) Evolution strategies—a comprehensve introduction. Nat Comput 1(1):3–53
5. Holland JH (1975) Adaptation in natural and artificial systems, Ann Arbor: The University of Michigan Press
6. Vose M (1999) The simple genetic algorithm: foundations and theory. MIT Press, Cambrige, Cambridge, MA
7. Sivanandam SN, Deepa SN (2008) Introduction to genetic algorithms, Springer, New York, USA
8. Haupt RL, Werner DH (2007) Genetic algorithms in electromagnetics, IEEE Press, Wiley-Interscience
9. Reeves CR, Rowe JE (2003s) Genetic algorithms: principles and perspectives. A guide to GA theory, Kluwer Academic Publishers, Dordrecht, The Netherlands
10. Yu X, Gen M (2010) Introduction to evolution algorithms, Springer, London, UK
11. KoLa JR (1992) Genetic programing, MIT Press, Cambrige, USA
12. Ladeh LA (1965) Fuzzy sets. Inf Control 8:338–353
13. Dubois D, Prade H (1980) Fuzzy sets and systems: theory and applications, Academic Press, New York, USA
14. Chen G, Pham TT (2001) Introduction to fuzzy sets, fuzzy logic and fuzzy control systems, CRC Press, Boca Raton, Florida, USA
15. Fuzzy Logic Toolbox (2019) User's guide, ver. 2.5, The Mathworks Inc.
16. Ross TJ (2004) Fuzzy logic with engineering applications, 2nd edn, Wiley, Chchester, UK
17. Buckley JJ, Eslami E (2002) An introduction to fuzzy logic and fuzzy sets, Springer-Verlag, Berlin
18. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biol 5(4):115–133
19. Caudil M, Butler C (1992) Understanding neural networks: computer exploration, vol 1, MA: The MIT Press, Cambrige
20. Hertz J, Krogh A, Palmer R (1995) Introduction to the theory of neural computation. Lecture notes, Santa Fe Institute, Adduson-WesleL Publishing Company
21. Hagan MT, Demuth HB, Beale MH, de Jesus O. Neural network design, 2nd edn, eBook
22. Micu DD, Czumbil L, Christoforidis GC, Simion E (2012) Neural networks applied in electromagnetic interference problems. Revue Roum Des Sci Tech Serie Electrotech et Energetique 57(2):162–171, ISSN: 0035-4066
23. Christodoulou C, Georgiopoulos M (2001) Applications of neural networks in electromagnetics, Artech House, Norwood, USA
24. Beale MH, Hagan MT, Demuth HB (2017) Neural networks toolbox. Users's Guide, ver. 10.0, the Mathworks Inc.
25. Şteţ D, Czumbil L, Micu DD, Ţopa V, Ancăş L () Stream gas pipeline in proximity of high voltage power lines. Part I—soil resistivity evaluation. In: 47th International universities' power egineering conference (UPEC), London, UK, September 4–7 2012
26. Czumbil L, Şteţ D, Micu DD, Ţopa V, Ancăş L (2012) Stream gas pipeline in proximitity of high voltage power lines. Part II—induced voltage evaluation. In: 47th Internatonal universities' conference on power energy(UPEC), London, UK, September 4–7 2012
27. Dawalibi FP, Donoso F (1993) Integrated analysis software for grounding, EMF and EMI. IEEE Comput Appl Power 6(2):19–24
28. Yang H, Yuan J, Long W (2001) Determination of three-layer earth from wenner four-probe test data. IEEE Trans Magn 37(5):3684–3687
29. Munteanu MS, Czumbil L, Micu DD, Braicu ŞF, Nemeti S, Pîslaru M (2017) Measurement of soil resistivity in order to determine the buried walls trajectory. Adv Electr Comput Eng (AECE) 17(1):103–108, ISSN: 1582-7445
30. CIGRE (1995) Guide on the infuence of high voltage AC power systems on metallic pipelines
31. Micu DD, Christoforidis GC, Czumbil L (2013) AC Interference on Pipelines due to Double Circuit Power Lines: A detailed study. Electr Power Syst Res 103:1–8

32. Gupta A, Thomas MJ (2006) Coupling of high voltage AC power line fields to metallic pipelines. In: 9th International conference on electromagnetic interference and compatibility, (INCEMIC), Bangalore, India, February 23–24 2006
33. Christoforidis GC, Labridis DB, Dokopoulos PS (2003) Inductive interference calculation on imperfect coated pipelines due to nearby faulted parallel transmission lines. Electr Power Syst Res 66(2):139–148
34. Christoforidis GC, Labridis DP, Dokopoulos PS (2005) A hybrid method for calculating the inductive interference caused by faulted power lines to nearby buried pipelines. IEEE Trans Power Deliv 20(2):1465–1473
35. zumbil L,Micu DD, Şteţ D, Ceclan A (2015) Inductive coupling between overhead power lines and nearby metallic pipelines. A neural network approach. Carpathian J Electr Eng (CJEE) 9(1):29–44
36. Micu DD, Christoforidis GC, Czumbil L (2012) Artificial intelligence techniques applied to electromagnetic interference problems between power lines and metal pipelines. In: Recurrent neural networks and soft computing, Intech, Ch 12, pp 253–274, Rijeka, Croatia
37. Micu DD,Czumbil L, Christoforidis GC, Ceclan A, Şteţ D (2012) Evaluation of induced AC voltages in underground metallic pipeline. COMPEL: Int J Comput Math Electr Electron Eng 31(4):133–1143
38. Czumbil L, Christoforidis GC, Micu DD, Şteţ D, Ceclan A, Pop O (2011) A user-friendly software application for induced AC interference evaluation. In: 46th International universities' power engineering conference, UPEC, Soest, Germany, September 5–8