A Neural-based Gradient Optimization in Large Power Systems

MAURY MEIRELLES GOUVÊA JR.
Polytechnic Institute
Pontifical Catholic University of Minas Gerais
Av. Dom José Gaspar, 500, Belo Horizonte
BRAZIL
maury@pucminas.br

Abstract: An artificial neural network (ANN) is commonly used as a universal function approximator which is very useful in several problems. The main ability of an ANN is to become similar to the behavior of the system being analyzed through examples acquired in past situations or through experiments. This paper presents a neural-based gradient optimization (NGO) method that applies an ANN in optimization problems in order to approximate the objective function to be minimized. According to this approach, the inputs of the ANN are the decision variables and the output is the objective function. Thus, an NGO uses the ANN topology to adjust the decision variables to find the optimal solution. The NGO is used to optimize a large power system without using an analytical model, and instead only use its historical record of behavior. The NGO was tested with the standard well-known power systems, the IEEE-14 and IEEE-118 bus. The performance of the NGO was compared with that of the traditional gradient-based optimization method.

Key–Words: System Optimization, System Identification, Optimal Reactive Power Flow, Artificial Neural Network.

1 Introduction

In order to solve optimization problems using traditional methods, e.g., descendant gradient [1], the mathematic model of the function to be optimized must be known, and it must be differentiable. Nevertheless, in several real problems, the function may not be differentiable, or its derivative may be difficult to compute. Thus, an optimization method that uses only computed values of the objective function is desired. Since an evaluation of that function normally requires a large computational effort, it is essential that the desired method can find the correct solution by using a small number of function evaluations.

There are some methods which solve optimization problems without derivatives, for example, by combining desired features of the bisection method and successive linear interpolation [2], the golden section search [3], successive parabolic interpolation [4], Powell's algorithm [5], and chaos optimization algorithm [6]. These methods may fail to satisfy at least one of the following requirements: there is no guarantee that the local minimum found is the global one and in certain applications where accuracy is not very important, a faster method is preferable.

An artificial neural network (ANN) [7] is a mathematical model inspired by the structure and functional aspects of biological neural networks. An ANN is commonly used as a universal function approximator which is very useful in several problems, such as, pattern recognition, nonlinear systems identification [15], control, and optimization applications [18, 19, 17, 16]. The main ability of an ANN is to become similar to the behavior of the system being analyzed through examples acquired in past situations or through experiments. Thus, an ANN becomes an input-output mapping of the systems.

This paper presents a neural-based gradient optimization (NGO) method that applies an ANN in optimization problems in order to approximate the objective function to be minimized. According to this approach, the inputs of the ANN are the decision variables and the output is the objective function. Since a system input-output map is built, by training, the ANN can emulate the objective function of the system. Thus, an NGO uses the ANN topology to adjust the decision variables to find the optimal solution, i.e., the method proposed replaces the original gradient vector.

In order to validate the method proposed, the NGO is used to optimize a power system. In this problem, the decision variables are the voltage control buses and the transformer taps, and the objective function is the sum of the load voltage deviations. Thus, the ANN can optimize the power system without using an analytical model, and instead only use its historical record of behavior. The NGO was tested with a standard well-known power system, the IEEE-118 bus bar. The performance of the NGO was compared with that of the traditional gradient-based optimization method.

The rest of this paper is organized as follows. Section 2 presents the formulation of the optimal power flow problem. Section 3 describes the ANN structure and training process. Section 4 presents the proposed neural-based gradient optimization method. Finally, Section 5 concludes the paper and suggests future lines of research.

2 Optimal Reactive Power Flow Problem

Industrial equipment used in power systems are designed for a given voltage level. If the voltage for these systems deviates from the reference value laid down, the performance of such items of equipment degrades and their life expectancies tend to drop. For instance, the torque of an induction motor is proportional to the square of the terminal voltage. Thus, the advantages of controlling the voltage level in power system are great, but this is performed in an interval which specifies the tolerance of voltage variations in the power system.

Moreover, the real losses in transmission lines depend on the real and reactive line power flow. Thus, it is possible to minimize these losses by selecting an optimum power flow with respect to real and reactive powers, in particular, because the reactive line power flow depends greatly on line-end voltage [20].

In a power system, the load buses (i.e., the buses in which the voltage is not controlled) throughout the system are designated and an attempt is made to maintain the voltage levels at specified values. This voltage control is achieved by acting in the following way [9]:

- Excitation control of generators, which maintains good voltage control at the generator buses;
- Switched shunt capacitors and/or reactors, which provide the capability of controlled reactive power injection into a bus;
- Tap-changing of transformers.

What motivates these control actions is the fact that the bus voltage is strongly related to the reactive power injection at the bus, i.e., to add reactive power means to increase the voltage.

The objective of the reactive power optimization is to minimize the voltage deviations of the set of numbers of load buses, S_{PQ}, which can be defined as follows

$$F(\mathbf{u}) = \sum_{i \in S_{PQ}} (V_i - V_i^{ref})^2, \tag{1}$$

subject to

$$\begin{aligned} P_{Gi} - P_{Di} &= V_i \textstyle\sum_{j=1}^{n} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}), \\ Q_{Gi} - Q_{Di} &= V_i \textstyle\sum_{j=1}^{n} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}), \end{aligned} \tag{2}$$

for $i = 1, \ldots, n$, and

$$\begin{aligned} & V_i^{\min} \le V_i \le V_i^{\max}, i = 1, \ldots, n \\ & T_i^{\min} \le T_i \le T_i^{\max}, i \in N_T \\ & Q_{Gi}^{\min} \le Q_{Gi} \le Q_{Gi}^{\max}, i \in N_G \\ & S_{ij} \le S_{ij}^{\max}, i, j = 1, \ldots, n, i \ne j, \end{aligned} \tag{3}$$

where $\mathbf{u}$ is the control variable vector $[V_g \, T_K]^T$, V_g, $g \in N_G$, are the generator voltage buses, T_k, $k \in N_T$, are the transformers, V_i and V_i^{ref} are the voltage magnitude and its reference value at i-th bus, P_{Gi} and P_{Di} are the real powers of generation and load, Q_{Gi} and Q_{Di} are the reactive power of generation and load at i-th bus, G_{ij} and B_{ij} are the real and imaginary parts of ij-th entry of bus admittance matrix, θ_{ij} is the voltage angle between buses i and j, S_{ij} is the power flow in the transmission line that connects buses i and j, T_i is the tap position of i-th transformer, N_G is the set of numbers of generator buses, N_T is the set of numbers of transformers, n is the number of buses, and the superscripts $\min$ and $\max$ are denote the corresponding lower and upper limits, respectively.

The set of $2n$ equality constraints, Equations (2), consists of the power flow equations (PFE) and the set of inequality constraints, Equations (3), is used to limit the bus voltages, transformer tap-setting, the reactive power flow installations, and the power flow in each transmission line.

3 Artificial Neural Networks

Learning is a feature of human beings in which while humans carry out similar tasks, they acquire the ability to enhance their performance. Researchers from several areas, such as computer science, engineering, psychology and neuroscience, have studied this feature in order to explain it. This field of science is normally called machine learning, where the learning can be classified into three categories: supervised, unsupervised, and reinforcement learning. Supervised learning needs a teacher to provide the input-output training patterns. The system adapts its parameters to generate a desired output pattern from a given input pattern. Whenever there is no teacher, i.e. there is unsupervised learning, a desired output for a given input pattern is not known, and the system has to adapt its parameters autonomously. In reinforcement learning, the system does not know the input-output patterns; thus, it only receives some feedback (reward or punishment) from its environment. The system uses this feedback signal provided from its actions to adapt its parameters.

An artificial neural network (ANN) is a training-based system inspired by the biological neural network. An ANN consists of artificial neurons fully connected by synaptic weights. Each neuron responds to stimuli, such as a biological neurons, and provides a response by means of an output signal. The information or knowledge of the ANN is stored in the synaptic weights. The learning of the ANN is performed in the training phase, where the weights are adjusted according to a knowledge base (a set of patterns) which represents the historical or behavioral pattern to which the ANN will approximate.

The artificial neuron was mathematically modeled by McCulloch and Pitts [10], which described a logical calculus for ANNs. McCulloch and Pitts showed that with a sufficient number of neurons and synaptic connections adjusted properly and operating synchronously an ANN would compute any computable function. Fifteen years later, Rosenblatt [11] created a new method for supervised learning, which is called the perceptron convergence theorem. After many years which saw only restricted further studies, Rumelhart et al. [12] created the backpropagation algorithm, the most popular learning algorithm for training, which became the most widely-use of the ANNs. Since then, research into and applications of ANNs have increased in different areas [8].

There are several models and paradigms of ANNs [7], for each of which there are recommended applications. In this study, the feedforward model [11] is used with the supervised training. For the method proposed, the problem to be optimized is simulated and a set of input-output patterns is formed. The ANN inputs are the decision variables and the output is the value of the objective function to be optimized.

The feedforward model consists of layers of neurons, each of them having its neurons fully connected to those of the next layer. Figure 1 shows an ANN with N inputs, N_h neurons in the hidden layer, and one output. The signals applied at the input layer are propagated throughout the network structure until the output layer. Thus, the ANN provides an output signal by the interactions of all neurons.

The signals which arrive in each neuron are weighted by coefficients, called synaptic weights; then, the weighted signals are summed by the neuron. Finally, the neuron output, is defined by an activation function which provides a high value if the weighted sum of the output is higher than a bias; otherwise, the output is not activated, and thus produces a low value. For an ANN with only one output, such that of Figure 1, the neuron output, y, can be defined as

$$y = \varphi(v), \tag{4}$$

where $\varphi(\cdot)$ is the activation function, which limits the amplitude of the output neuron, and

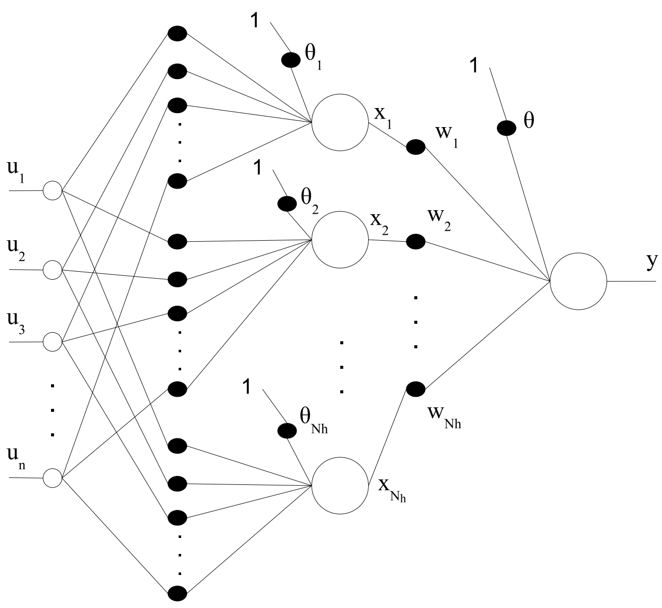

Figure 1: Model of a feedforward neural network model

$$v = \sum_{i=1}^{N_h} w_i\, x_i - \theta, \tag{5}$$

where w_i is the i-th synaptic weight of the output neuron, θ is the bias of the output neuron, N_h is the number of neurons of the hidden layer, and

$$x_i = \varphi_i(v_i) \tag{6}$$

is the output of the i-th neuron of the hidden layer, where $\varphi_i(\cdot)$ is the activation function of

$$v_i = \sum_{j=1}^{n} w_{ij}\, u_j - \theta_i, \tag{7}$$

where w_{ij} and θ_i are the j-th synaptic weight and the bias of the i-th neuron of the hidden layer, and u_j is the j-th ANN input.

Normally, the normalized amplitude range of the neuron output is the closed unit [0,1] interval or [-1,1]. The activation function used in this study, sigmoid function, is the most commom form of activation function used in ANNs. The sigmoid function is an increasing function that shows a balance between linear and nonlinear behavior [13]. The sigmoid function used in this study is the logistic function [14], defined as follows

$$\varphi(v) = \frac{1}{1 + \exp(-k\, v)}, \tag{8}$$

where k is the slope parameter of the logistic function. By varying k, it obtains logistic functions with different slopes for $k = \{0.4, 1.0, 2.0\}$, as shown in Figure 2.

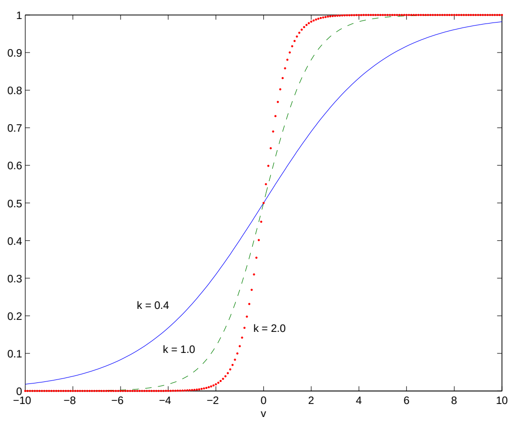

Figure 2: Logistic functions with several slopes

3.1 Training of Artificial Neural Networks

The application of ANNs greatly increased after the backpropagation training algorithm [12] was introduced. ANN training, also called ANN learning, is performed by using samples (set of patterns) taken from the environment represented by input-output pairs. In the learning process, the ANN adapts its parameters by means of an algorithm to generate the output patterns desired from given input patterns. The set of patterns is defined by

$$\Psi = \{(\mathbf{u}_i, \mathbf{d}_i)\}, i = 1, \ldots, P \tag{9}$$

where $(\mathbf{u}_i, \mathbf{d}_i)$ is i-th pattern, $\mathbf{u}_i$ is the i-th input vector and $\mathbf{d}_i$ is the i-th output vector desired. In ANN training, a signal is applied and one output signal is obtained; if this output is other than $\mathbf{d}_i$ there is an error and the synaptic weights must be adjusted. The backpropagation algorithm deals with the training as a non-constrained global optimization problem, where the quadratic error function, to be minimized, is defined by

$$E = \frac{1}{2} \sum_{i=1}^{m} (d_i - y_i)^2 \tag{10}$$

where y_i^d and y_i are the i-th desired and current ANN outputs, and m is the number of ANN outputs. The patterns are presented sequentially until the existing difference between the ANN output and the desired output, considering all P patterns, is greater than an error. The adjustment of the j-th weight of the i-th neuron is defined by

$$\Delta w_{ij} = -\eta\, \delta_i\, x_i, \tag{11}$$

where the minus sign accounts for gradient descent in weight space, i.e., following a direction for weight changes that reduces E, Equation 10, and

$$\delta_i = \varphi'(v_i)\,(d_i - y_i) \tag{12}$$

for the i-th output layer, and

$$\delta_i = \varphi_i'(v_i) \sum_j \delta_j\, w_{ji} \tag{13}$$

for the i-th neuron of a hidden layer, where the index j refers to neurons in the next layer in which the i-th is connected. The η coefficient, called the learning rate, determines the length of the step under the quadratic error function.

4 Neural-Based Gradient Optimization

The ANN used in this methodology has two specific goals: to emulate the system, a model for which is not available, and to optimize the performance of this system using its own neural topology. In the optimization problem, the neural network inputs represent the decision variables and the neural network outputs represent the objective functions. The neural network output can be described as follows

$$y = F(\mathbf{u}), \tag{14}$$

where y stands for the ANN output, that is, the objective function, and $\mathbf{u}$ is the ANN inputs, that is, the decision variables.

In a minimization problem, in which the descendant gradient method is used, the decision variable adjustments, for each ANN output that represents an objective function, are made as follows

$$\mathbf{u}(k+1) = \mathbf{u}(k) - \eta\, \nabla F(k), \tag{15}$$

where $\mathbf{u}(k)$ is the decision variable vector in the k-th iteration, $\nabla F(\cdot)$ is the gradient vector of the $F(\cdot)$ objective function, and η is the step under $F(\cdot)$. The adjustment of the i-th decision variable is made as follows

$$u_i(k+1) = u_i(k) - \eta\, \frac{\partial F}{\partial u_i}. \tag{16}$$

The partial derivation of the ANN output, Equation (14), with respect to the decision variable u_i is not direct, because y is a function of v, Equation (4). Thus, the differential of output y can be expressed as

$$dy = \varphi'(v)\, dv, \tag{17}$$

and the differential of v, Equation (5), can be expressed as follows

$$dv = \sum_{i=1}^{N_h} w_i\, dx_i. \tag{18}$$

Up to this point, the differential of output y can be expressed as follows

$$dy = \varphi'(v) \sum_{i=1}^{N_h} w_i \, dx_i, \tag{19}$$

where

$$dx_i = \varphi_i'(v_i) \, dv_i, \tag{20}$$

is the differential of the output x_i of the hidden layer neuron, Equation (6), and v_i, Equation (7), is a function of $u_1, u_2, \ldots, u_n$, which corresponds to the ANN inputs, for an ANN such that of Figure 1. Thus, the differential dv_i with respect to the j-th input can be expressed as follows

$$dv_i = w_{ij} \, du_j. \tag{21}$$

Finally, replacing Equations (20) and (21) into Equation (19), the differential of output y results in

$$dy = \varphi'(v) \sum_{i=1}^{N_h} w_i \, \varphi_i'(v_i) \, w_{ij} \, du_j. \tag{22}$$

The partial derivation of the objective function, F, i.e., the ANN output y, with respect to the j-th decision variable, u_j, results in

$$\frac{\partial F}{\partial u_j} = \varphi'(v) \sum_{i=1}^{N_h} w_i \, \varphi_i'(v_i) \, w_{ij}. \tag{23}$$

The derivation of function $\varphi(\cdot)$ with respect to v is

$$\varphi'(v) = \frac{k \, e^{-kv}}{(1 + e^{-kv})^2}. \tag{24}$$

Thus, after training, the neural network is used, in real time, for emulating the system and adjusting the decision variables to optimize the system. Thus, the gradient vector of neural topology, instead of the gradient vector of the system original model, is used to adjust the decision variable. This process is followed for each objective function which is represented by ANN output. The main advantage of NGO is that the gradient method can be used in problems in which the gradient vector is unavailable. This is possible by simulating the system and building a set of samples which describes the behavior of the system. Thus, this set is used to training the ANN and NGO can be used to optimize the performance of the system.

5 Experimental Studies

This section presents studies which used the IEEE-14 and IEEE-118 bus systems. In order to adapt the power system variables to the neural network domain, a linear normalization was applied to those used as neural network inputs and outputs. The validation of the method proposed was performed by experiments with a gradient-based algorithm [1].

The IEEE-14 bus system, Figure 3, represents a portion of the American Electric Power System (in the Midwestern US) as of February, 1962. The system consists of 5 synchronous machines, 3 of which are synchronous compensators used only for reactive power support. This system is widely used for voltage stability and low frequency oscillatory stability analysis. In the IEEE-14 bus, a three layer MPL with the backpropagation training algorithm was used. The controlled buses (PV) were used as the four neural network inputs. The neural network output was the sum of load bus (PQ) deviations from the reference voltage, Equation (1). Table 1 shows the optimization results after the ANN training. The second and third columns show the real power losses from gradient-based (GB) and neural-based gradient optimization (NGO) optimization methods, and the fourth and fifth columns show the PQ voltage deviations from GB and NGO methods.

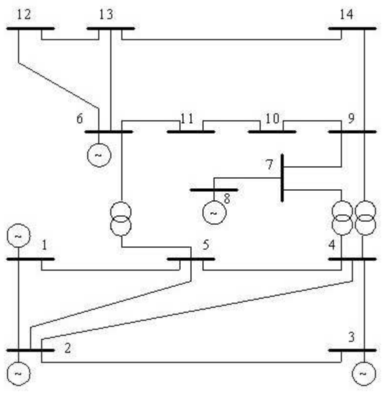

Figure 3: IEEE-14 Bus System

GB and NGO spent 4 iterations to stop the optimization process. In both real power loss and PQ voltage deviation, GB and NGO reached similar results. The main purpose of the NGO method was reached, i.e., the trained ANN acquired sensitivity to the objective function with respect to the decision variables. This claim could be concluded because the PQ voltage deviation by NGO method decreased at each iteration, Table 1.

The IEEE-118 bus system, Figure 4, represents a part of the American Electric Power System (in the Midwestern US) as of December, 1962. The system consists of 41 synchronous generators and 27 synchronous compensators with 186 branches. In this system, a three layer MPL with the backpropagation training algorithm was used. The controlled buses (PV) and the regulating transformers were used as the

Table 1: Results of IEEE-14 system optimization

	Real Power Loss		PQ Voltage Deviation	
Iter.	GB	NGO	GB	NGO
0	7.91	7.91	0.6022	0.6822
1	7.81	7.91	0.4704	0.5301
2	7.80	7.92	0.3675	0.4075
3	7.80	7.92	0.2876	0.3281
4	7.80	7.93	0.2277	0.2549

neural network inputs. Regarding the size and complexity of this system, several neural network topologies were trained. The neural network output was the sum of load bus (PQ) deviations from the reference voltage, Equation (1). The following subsections describe the ANN training and the optimization process for the IEEE-118 bus system.

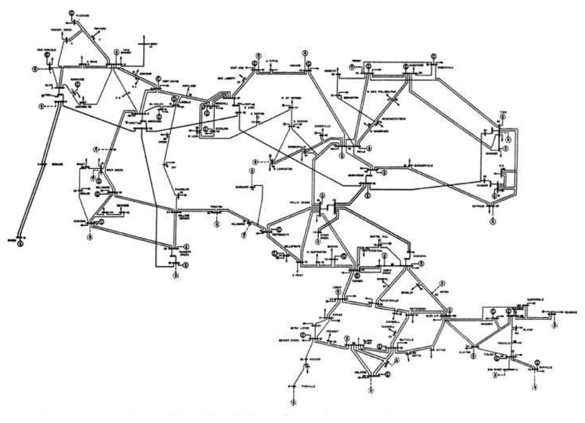

Figure 4: IEEE-118 Bus System

5.1 Neural Network Training

Given the size and complexity of the IEEE-118 bus bar system, several rounds of training with different topologies were performed in order to find, empirically, the best one. The main performance criterion was validation using the mean square error (MSE). The set of examples consisted of 5,000 training examples, 1,000 validation examples, and 1,000 examples to test the artificial neural network (ANN).

All training was performed in 1,000 epochs in order to find the best NN weights, i.e., those which provide the minimum validation error. For this purpose, whenever a configuration of weights reached a minimum validation error, these parameters were saved.

Table 2 shows the MSE of the set of training, validations, and tests for all topologies (Top.). The second and third columns show the mean and final MSEs from the set of training, the fourth to sixth columns present the mean MSE, standard deviation (SD), and minimum MSE from the set of validations. In the sixth column, the epoch in which the minimum MSE was reached is shown in brackets. The seventh column shows the MSE from the set of test (after training).

The difference between the mean and final MSE of the training set, the second and third columns in Table 2, is important as this shows the stability of the weight adjustments, i.e., the smaller this difference was, the faster the backpropagation algorithm reached an MSE value similar to the final one. A topology with this feature is useful for on-line training, because it may converge to a promising set of weights as fast as possible. The mean MSE, the SD, and the minimum MSE of the validation set indicate what the performance of the ANN is with respect to examples not presented in the weight adjustment process. The SD shows how the MSE varied with respect to the weight adjustments, i.e., the lower the SD, the smaller the impact of the weight adjustments was. Of course, SD analysis must be associated with the mean MSE from the validation set. A low mean MSE demonstrates a good ability to generalize capacity when this value is similar to the minimum one (sixth column). Ideally, the SD must be low and the minimum MSE must be close to the mean one. This shows that the ANN may achieve a low error rate almost throughout the training process. The MSE of the set of tests (last column) is also a very useful performance indicator. Ideally, the test error should be close to the minimum MSE of the validation set. In other words, this indicates that the set of weights reached the level of showing its ability to generalize for unknown examples is good.

Figure 5 shows the MSE of the training and validation sets for topologies with only one hidden layer of each of 5, 20, 50, and 80 neurons. The profiles of Figures 5(a)-(d) reinforce the analysis above, from Table 2. Figures 5(a)-(b) present two stable topologies as can also be seen in the lines of the topologies with one hidden layer of 5 and another of 20 neurons (low SD and minimum MSE close to the mean). Nevertheless, the topology with hidden layer of 20 neurons, Figure 5(b), reached the smallest MSE of the validation set. Table 2 and Figure 5 show that topologies in which the hidden layer had 40 or more neurons decrease the training performances. With these topologies, all per-

Table 2: MSE and standard deviation (SD) ($\times 10^{-3}$) for IEEE-118 bus bar system

	Training		Validation			
Top.	Mean	Final	Mean	SD	Min.	Test
5	3.5337	3.4300	4.0615	0.3236	3.9370 (231)	3.7487
10	2.3223	2.1450	3.3713	0.3873	3.1650 (088)	3.3949
20	1.9334	1.4910	3.5356	0.3764	2.9850 (157)	3.3417
30	1.6583	1.2200	4.1310	0.4427	3.3830 (076)	4.5169
40	2.0137	1.5980	5.0917	0.8992	3.6400 (124)	6.1511
50	2.0249	1.3880	5.7870	0.9739	3.5420 (093)	6.6508
60	2.3069	2.1680	6.7219	1.4171	4.7270 (156)	8.8947
80	8.8512	3.3580	14.9753	5.7790	5.3110 (028)	11.7778
100	25.8993	4.8640	29.3427	19.0055	5.8020 (019)	11.0851

formance criteria became worse (see the mean MSE of the validation set, and the SD and MSE of the set of tests). Starting with the topology with a hidden layer of 50 neurons, Figure 5-(c), the MSE of the validation set increased quickly after its minimum value was reached (epoch 93 for the hidden layer with 50 neurons). The topologies with one hidden layer of 80 and another of 100 neurons (see Figure 5-(d) for the former) were unstable and all their MSEs were very high. Figure 5-(d) shows that the topology with the hidden layer of 80 neurons was very sensitive to the weight adjustments. Thus, the MSEs of the training and validation sets varied greatly throughout training.

Given the analysis above, it is possible to conclude, especially from Table 2, the performances of the topologies with hidden layers of 10, 20, and 30 neurons were similar to each other. The topology chosen to simulate the IEEE-118 system was that with a hidden layer of 20 neurons, because it reached the smallest MSE in the validation and test sets (sixth and seventh columns). This topology reached its lowest MSE in the validation set at epoch 157 – see sixth column in Table 2. For this optimum set of weights, deemed the best ANN topology, the MSE of the test set was 3.3234 $\times 10^{-3}$.

Table 3: The topology of the best ANN in the test set

Error	Magnitude
Mean absolute error	0.0632
SD of absolute error	0.0515
Maximum absolute error	0.5021

Table 3 shows the results obtained by the best ANN topology given a test set with 1,000 examples. The maximum absolute error was high, 0.5021 (50.21%); nevertheless, as can be seen in Figure 6, it was only this error that reached a high value. Almost all of the absolute errors were smaller than 0.1 (10%), as can be seen from the mean and SD of the absolute error in Table 3. This claim is also reinforced in Figure 7 which presents a normal distribution of the error of the test set and confirms that almost of the errors were close to zero. For a large and complex system, such as the IEEE-118 bus bar, these results may well be acceptable. Thus, the topology with hidden layer of 20 neurons was approved in the training and can be used to simulate the IEEE-118 system.

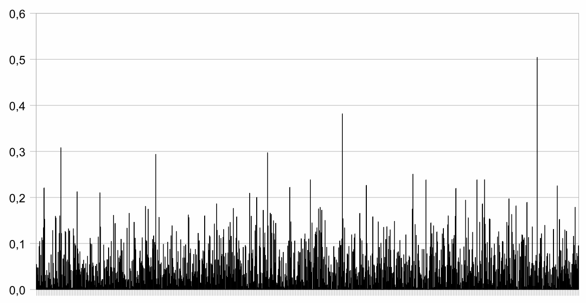

Figure 6: Absolute error bar of the test set for the topology of the best ANN

5.2 Simulating and Optimizing the IEEE-118 System

This section presents the experiment with the NGO and the descendant gradient method [1], in which the target is to minimize the PQ bus deviations from a reference value. For both methods, the constant η, Equation (15), was set to 0.5. The real power loss was also considered when analyzing the results. The stop criterion was that the mean variation of the objective function at the last three iterations is less than 1%. The reference voltage for the PQ buses was 1 pu.

Table 4 shows the results of the experiment. The second and third columns present the real power losses from the gradient-based (GB) and the neural-based gradient optimization (NGO) methods. The

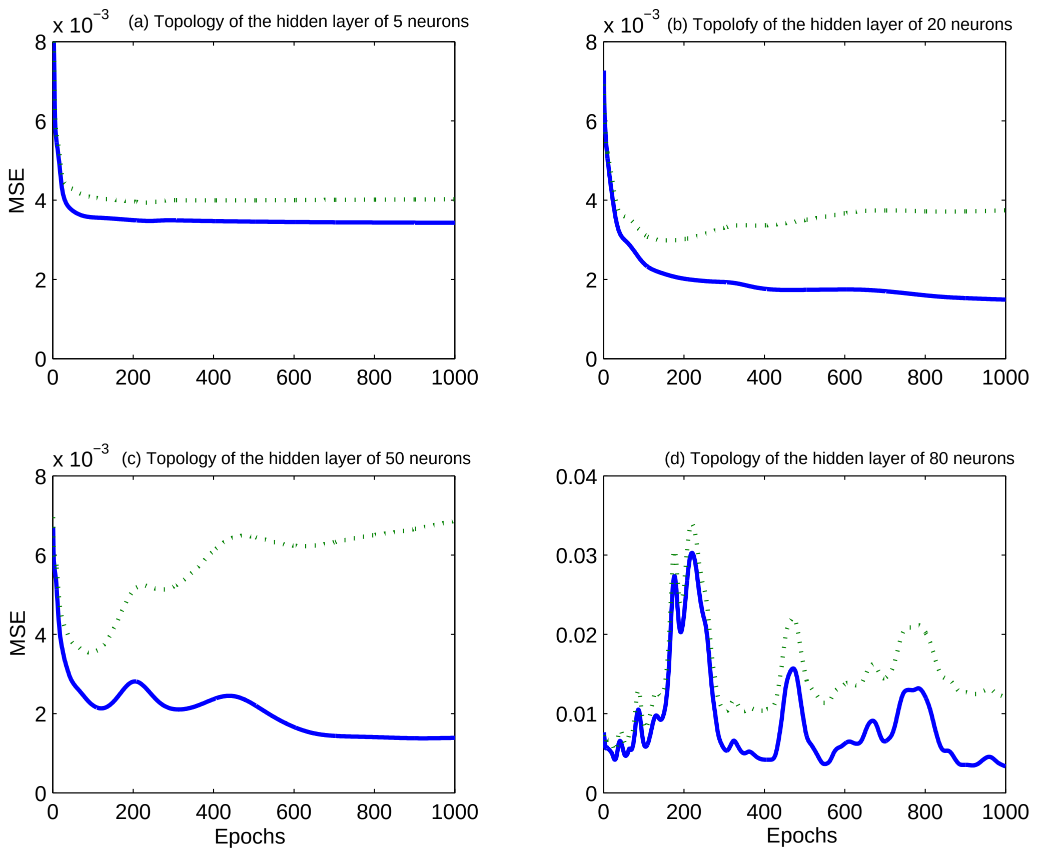

Figure 5: MSE of the set of training (solid line) and validation (dotted line)

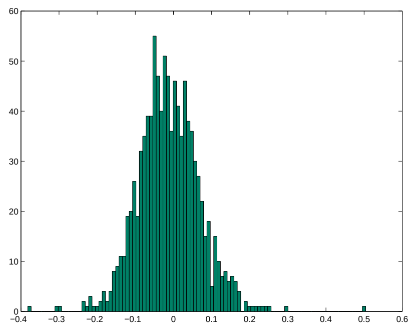

Figure 7: Error distribution of the test set for the topology of the best ANN

fourth to sixth columns present the sum of PQ voltage deviations: column 4 by GB, column 5 by PFE in the NGO after adjusting the decision variables, and column 6 by the ANN in the NGO.

The GB spent only three iterations to stop the optimization process and the NGO stopped it after seven iterations. Given that the objective of the optimization process, the PQ deviations, NGO reached the best result, with a value 43% smaller than that of the GB. The real power loss from NGO was 1.83% larger than that of the GB method. This was an acceptable result because more energy dispatch was necessary to decrease the PQ deviations.

Another important result was the difference between the system output, column 5, and the ANN output, column 6, in Table 4. Although the NGO reached a promising result, the sum of PQ deviations produced by the ANN was very different from the real one, column 5 in Table 4. It may be argued that the set of training examples was not representative enough. Nevertheless, as to main purpose of the NGO the objective was reached, i.e., the trained ANN acquired sensitivity to the objective function with respect to the decision variables. For this objective, the set of examples was enough. To provide the sum of PQ deviations as close as possible to the real one, the size of the set of examples must be increased.

6 Conclusions

This paper presented a method that uses an artificial neural network (ANN) for modeling and optimizing complex systems. The new neural-based gradient optimization (NGO) method employs an ANN to replace the analytic modeling system with one that simulates and trains. Thus, the NGO deals with the topology of a trained ANN for optimization purposes, by replacing the gradient vector of the original system.

Table 4: Results of IEEE-118 system optimization

	Real Power Loss		PQ Voltage Deviation		
Iter.	GB	NGO	GB	Sys. Output	ANN Output
0	158.11	158.11	0.0529	0.0529	0.0892
1	157.32	167.51	0.0388	0.0258	0.0761
2	157.64	164.17	0.0324	0.0210	0.0723
3	157.97	162.09	0.0275	0.0184	0.0704
4	—	159.31	—	0.0169	0.0690
5	—	159.23	—	0.0162	0.0682
6	—	159.40	—	0.0158	0.0673
7	—	160.10	—	0.0157	0.0666
8	—	160.86	—	0.0156	0.0659

The IEEE-14 and IEEE-118 bus systems were used to evaluate the performance of the method proposed. The voltage deviation, Equation (1), was used as the objective function to be minimized. Several training procedures with different ANN topologies were performed and analyzed in order to choose the best one. The experiments to minimize the load voltage deviations of the systems were conducted to test and discuss the performance of the method proposed. In the experiment with the IEEE-118 bus system, the NGO outperformed the classic gradient-based method.

In the experiments, NGO tended to minimize the objective function and reached a value smaller than that of the gradient-based method. NGO needed more iterations to stop the search than the gradient-based method; nevertheless, the processing time spent by an ANN is very low. Another important result was that the ANN output error (the voltage deviations) in the experiments was higher that of the set of test – see the last two columns in Table 4. This problem was probably caused due to the size of the set of training and the complexity of the IEEE-118 system. It is possible that by increasing this size, the errors will tend to decrease. Nevertheless, this fact is not enough to affect the NGO performance adversely, because even when the output errors were higher than those expected, the ANN acquired sensitivity to the system. The suggestion for this claim is that NGO followed the path to the (local/global) minimum – see the *Sys. Output* column in Table 4.

Given these results, the supposition may be made that ANN topology can be applied to other types of optimization problems, where the gradient vector is difficult to calculate or needs a high computational effort. In future research studies, the ANN topology might also be used in hybrid systems, for example, combining an NGO with simulated annealing in order to create a stochastic search algorithm.

Acknowledgements: The author gratefully acknowledge the support received from the Pontifical Catholic University of Minas Gerais for this research study.

References:

[1] D. A. Pierre. *Optimization Theory with Applications*. Dover, Mineola, NY, 1986.

[2] A. Kaw and E. Kalu. *Numerical Methods with Applications*. http://www.autarkaw.com, 2 edition, 2010.

[3] J. S. Arora. *Introduction to optimum design*. Elsevier Academic Press, San Diego, CA, USA, 2004.

[4] P. Jarratt. An iterative method for locating turning points. *Computer Journal*, (10):82–84, 1967.

[5] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, (7):152Ű-162, 1964.

[6] M. S. Tavazoei and M. Haeri. An optimization algorithm based on chaotic behavior and fractal nature. *Journal of Computational and Applied Mathematics*, (206):1070Ű-1081, 2007.

[7] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, second edition, 1998.

[8] M.R.G. Meireles, P.E.M. Almeida, and M.G. Simões. A comprehensive review for industrial applicability of artificial neural networks. *IEEE Transactions on Industrial Electronics*, 50(3):585–601, 2003.

[9] O. I. Elgerd. *Electric Energy Systems Theory*. Mcgraw-Hill, New York, 2 edition, 1982.

[10] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[11] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[12] D. Rumelhart, G. Hinton, and R. Williams. *Learning Internal Representations by Error Propagation*. MIT Press, Cambridge, MA, USA, 1986.

[13] A. K. Mennon, C. K. Mohan, and S. Ranka. Characterization of a class of sigmoid functions with applications to neural networks. *Neural Networks*, 9:819–835, 1996.

[14] W. Feller. *An introduction to Probability Theory and its Applications*, volume 1. John Wiley, New York, third edition, 1968.

[15] F. Neri. Agent Based Modeling Under Partial and Full Knowledge Learning Settings to Simulate Financial Markets. *AI Communications*, 25(4):295–305, 2012.

[16] A.M. Kassem. Neural control design for isolated wind generation system based on SVC and nonlinear autoregressive moving average approach. *WSEAS Transactions on Systems*, 11(2):39–49, 2012.

[17] L. Roseiro, U. Ramos, R. Leal. Genetic algorithms and neural networks in optimal location of piezoelectric actuators and identification of mechanical properties. *WSEAS Transactions on Systems*, 5(12):2911–2916, 2006.

[18] M.M. Gouvêa Jr., L.D.B. Terra. Fuzzy Goal Programming Applied to the Optimal Reactive Power Flow Problem. *Proceedings of IEEE Budapest Power Tec'99*, Budapest, 1999.

[19] M.M. Gouvêa Jr., L.D.B. Terra. Neuro-Fuzzy Hybrid System Applied to the Optimal Reactive Power Flow Problem. *WSEAS International Conference on Simulation, Modelling and Optimization*, Skiathos, 2002.

[20] G.V.S.K. Rao, K. Vaisakh. Optimal allocation of real and reactive power for loss and marginal cost reduction. *WSEAS Transactions on Systems*, 2006.