Estimation of the steady states parameters in open-loop distribution networks based on feedforward neural networks

Muzaffar Khudayarov[1,2*], and *Bahrom* Bobonazarov[3]

[1]Tashkent State Technical University named after Islam Karimov, Department of Power Plants, Networks and Systems. 100095, 2 University Street, Tashkent, Uzbekistan
[2]Department of Power Supply and Renewable Energy Sources, "Tashkent Institute of Irrigation and Agricultural Mechanization Engineers" National Research University, 100000 Tashkent, Uzbekistan
[3]Karshi Engineering Economic Institute, Faculty of Engineering, 180101, Qashqadaryo region, Qarshi, 225 Mustaqillik shoh Street, Karshi, Uzbekistan

Abstract. Power flow calculations play major role during the operational stage of any distribution networks for its control, as well as during the design stage. Moreover, the main purpose of any power flow calculations is to compute precise steady-state voltages of nodes, the real and reactive power flows on each branches, under the assumption of known loads. That is, one of the main results of the calculation is steady-state voltages of nodes. As a rule, iterative methods are used to calculate load flows in distribution networks. This places high demands on calculations in terms of speed and reliability of obtaining results in any operating conditions. Given this in the article presents models for node voltages estimation in distribution networks based on feedforward artificial neural networks. Their use makes it possible to increase the speed of the power flow calculations in distribution networks. We examined the effectiveness of the models on the example of real schemes of 6-10 kV open-loop distribution networks.

1. Introduction

As you know, Power Flow Analysis is the computational procedure required to determine the steady state operating parameters of a distribution network (DN) from the given line data and bus data [1]. The steady state operating parameters are the voltages on nodes, currents, active and reactive powers and losses on each branches. There are several iterative methods for power flow calculations, such as backward/forward, Gauss-Seidel, and Newton's method [2]. The power flow analysis commonly used in distribution networks includes backward/forward method. Calculation by iterative methods is associated with great mathematical and computational difficulties and they are ineffective for operational calculations, due to the large time spent. Therefore, finding the faster power flow calculation methods is an important task. The development of methods of computational mathematics and information technology makes it possible to improve the operational control of distribution networks. At present, it is promising to calculate power flow parameters based on artificial neural networks (ANN) [3,4,5].
Therefore, this article deals the problem of estimating the power flow parameters using a feedforward ANNs. Comparative evaluation of the results is also performed. Perceptron model (created by the *fitnet* function) and cascade network (created by the *cascadeforwardnet* function) are used for modeling. The modelling tool is MATLAB and its library Neural Network Toolbox.

2. Methods

2.1. Backward/forward method

In most cases, when performing power flow calculations in distribution networks, the average loads (Eq.1, 2) are used as the base. The calculation of the average loads at the nodes is performed according to the power deliveries (W_{Pj}, W_{Qj}) for the billing period T:

[*]Corresponding author: muzaffar_hb@mail.ru

$$P_j = \frac{W_{Pj}}{T}, \qquad Q_j = \frac{W_{Qj}}{T} \tag{1, 2}$$

Average node loads P_j, Q_j, and reference voltages U_{GU} are used to calculate the power flow parameters in the distribution network. In this case, the calculation involves two computation processes at each iteration.
The backward process (first stage) involves the power flow solutions starting from the branch of the end nodes moving toward the branch connected to the head node (Eq.3):

$$\dot{S}_{ij} = \sum_{k \neq i}^{n_j} \dot{S}_{jk} + \dot{S}_j + U_j^2 \dot{Y}_{shj} + \frac{1}{2} \dot{Y}_{cij} \left(U_i^2 + U_j^2 \right) + \Delta \dot{S}_{ij} \tag{3}$$

where: n_j is the number of branches connected to node j; S_j - load at node j; U_i, U_j - voltage modules at nodes i and j; Y_{shj} - conductivity of the shunt at node j; Y_{cij}-capacitive conductance of the *i-j* branch; ΔS_{ij} - power losses in the *i-j* branch, are determined by the expression (Eq.4):

$$\Delta \dot{S}_{ij} = \Delta P_{ij} + j \Delta Q_{ij} = \frac{\left(\sum_{k \neq i}^{n_j} \dot{S}_{jk} + \dot{S}_j + U_j^2 \dot{Y}_{shj} + \frac{1}{2} \dot{Y}_{cij} U_j^2 \right)^2}{U_j^2} \dot{Z}_{ij} \tag{4}$$

In the forward process (second stage) calculates the voltage at each node starting from the reference node to the end nodes, i.e. from the beginning to the end of the network (Eq.5):

$$\dot{U}_j = U_j' + jU_j'' = \left[U_i - \frac{P_{ij} R_{ij} + Q_{ij} X_{ij}}{U_i} + j \frac{P_{ij} X_{ij} + Q_{ij} R_{ij}}{U_i} \right] \cdot K_{ij} \tag{5}$$

where, $K_{ij} = U_j/U_i$ if the branch is transformer and $K_{ij} = 1$ if the branch is linear.
The initial values of the node voltages are considered the same and equal to the nominal voltage of the network.
After each iteration, the power flow convergence is tested. The stopping criteria are the convergence of voltage obtained by tracking the differences of voltage between two successive iterations. The algorithm stops when the conditions in (Eq.6) are met.

$$max \sqrt{\left(U_j'^{(k+1)} - U_j'^{(k)} \right)^2 + \left(U_j''^{(k+1)} - U_j''^{(k)} \right)^2} \leq \varepsilon, \tag{6}$$

where k is the iteration number; $\varepsilon = 0.001$ is the specified accuracy of calculating the voltage mode; $j = 1,2 \ldots, n$, - the number of DN nodes.
If condition (6) is not satisfied, then proceed to calculations (3) ÷ (5) of the next iteration.
The results of power flow calculations are the voltage of the nodes U_i, the powers of the branches P_{ij}, Q_{ij} and the head node P_{GU}, Q_{GU}, the power losses in the branches ΔP_{ij}, ΔQ_{ij}, and total power losses in the network ΔP_{sum}, ΔQ_{sum}.

2.2. Method based on ANN

To estimate the node voltages Ui we can represent the ANN model as an approximating function (Eq.7):

$$U_i = \sqrt{U_i'^2 + U_i''^2} = F(U_{GU}, P_j, Q_j), \tag{7}$$

where $i = 1,2, \ldots, n$ - DN nodes, j - load nodes; P_j, Q_j - average loads; U_{GU}- head node voltage; U_i - DN node voltages.
To estimate of the DN steady-state node voltages and a comparative analysis of the results of calculations are considered feedforward ANN.
In a feedforward ANN with one or more hidden layers, the information moves in only one direction from the input through the hidden layers to the output and do not form a cycle. In this case, two types of feedforward ANN are considered: a perceptron and a cascadeforward network. The perceptron is the simplest type of ANN. Unlike the perceptron, cascadeforward network have connections from each next layer to all previous layers.
The construction of both types of neural networks involves the following steps:

1. Generation of statistical data and breakdown into training, testing and chacking samples.

2. Selecting the ANN architecture.
3. Training ANN on training sample.
4. Evaluation of ANN adequacy on test sample.

Next, the best ANN model from the considered ones is selected. The choice of the best ANN is performed according to the cheking sample data.

3. Results and Discussions

3.1. Generation of statistical data and breakdown into samples

We generate statistical data by simulating loads in the presence of a DN scheme. In this setting, the simulation algorithm consists of the following two stages:
- Simulation of loads by the Monte Carlo method [6] (obtaining a cross section for the implementation of random processes describing the load for each half hour of the design period-month);
- Power flow calculation of the DN by backward/forward method for a given moment.

To implement this algorithm, we have selected a number of 6-10 kV distribution network schemes. As an example, consider a 10 kV DN with 10 nodes and 4 loads (Figure1).

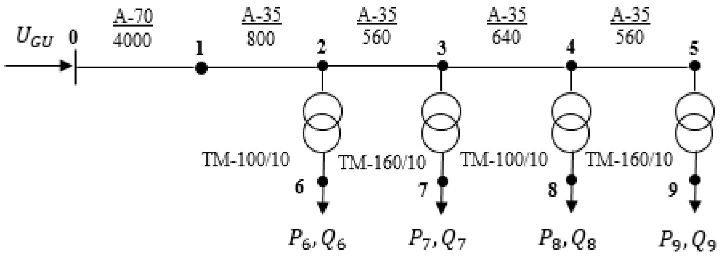

Fig. 1. Operational schema of the 10 kV distribution network

We simulate loads by changing the following parameters in the specified ranges:

Head node voltage $U_{GU} = 9.5 \div 10.5$ kV;

Load factor of transformers $k_z = 0.1 \div 0.8$;

Head node power factor $cos\varphi = 0.7 \div 0.9$;

Further, according to the received loads and head node voltage, we perform the power flow calculation by a backward/forward method. In total, we perform 1488 calculations (for each half hour during the month).

Based on the simulation data and the calculation results, a sample was formed consisting of 1488 pairs of "inputs-outputs" (I / O) statistical data. The inputs are the head node voltage, the active and reactive power of the loads (U_{GU}, P_6, Q_6, P_7, Q_7, P_8, Q_8, P_9, *and* Q_9), and the outputs are the node voltages (U_1-U_9), i.e. the approximation function has the following form (Eq.8):

$$[U_1^k, U_2^k, \ldots, U_9^k] = F(U_{GU}^k, P_6^k, Q_6^k, P_7^k, Q_7^k, P_8^k, Q_8^k, P_9^k, Q_9^k), \quad (8)$$

where k is the ordinal number of statistical data.

The resulting database is divided into training and test samples. We use a training set for adjusting synaptic coefficients, which includes 1042 pairs, i.e. 70% of the I/O data. The test sample includes 446 pairs or 30% of the I/O data that are not involved in the training process and serves to check the quality of training for each ANN.

In addition, we form a checking sample, which we use to select the best model from the generated ones. This sample includes 366 I/O data pairs (for each half hour during the week) by modelling loads by changing the following parameters in the specified ranges:

Head node voltage $U_{GU} = 9.5 \div 10.5$ kV;
Load factor of transformers $k_z = 0.8 \div 0.85$;
Head node power factor $cos\varphi = 0.9 \div 0.99$.

As can be seen from the above, the training and test samples refer to one general sample, and the checking sample differs from them and belongs to another.

3.2. Selecting the architecture and training of artificial neural networks

When designing ANN models, one of the problems is the choice of its architecture. To solve this problem, there are no clear rules either for the choice of the number of hidden layers, or for the choice of the number of neurons in the layers. The choice of network architecture is based on the experience of the researcher. At the same time, there are a number of general recommendations for choosing the number of layers based on the Kolmogorov-Arnold-Hecht-Nielsen theorem [7, 8 and 9]:

- If a function is defined on a finite set of points, then a three-layer perceptron is able to approximate it.
- If a function is continuous and defined on a compact region, then a three-layer perceptron is able to approximate it.
- The rest of the functions that can be trained in neural networks can be approximated by a four-layer perceptron.

Thus, theoretically, at most four layers are required, i.e., with two hidden layers. But at the same time, one hidden layer is enough to solve most technical problems.

An overview of methods for choosing the number of neurons in the hidden layer is given in [10]. These methods can be divided into analytical [11,12] and constructive [13,14]. Analytical methods require the existence of some mathematical formulas to estimate the number of neurons in the hidden layer. For example, in [11], the formula $N_{hid} \geq 2n + 1$ is presented, where n is the number of inputs of ANN. In constructive methods, the number of neurons in the layers is determined in two ways: reduction and augmentation.

In the reduction algorithm, the number of neurons in the hidden layer is excessive. Then they decrease by excluding the least significant neurons. In the augmentation algorithms, there is one neuron in the hidden layer, and then they are sequentially increased. This method is more appropriate for choosing the number of neurons in the hidden layer.

The number of neurons in the output layer depends on the amount of output data and we set it as a vector.

After choosing the network architecture, we carry out training of the neural network, i.e. adjust the weights and threshold. In particular, when using a feedforward ANN, this is the backpropagation (BP) algorithm. There are also variations of this algorithm, e.g., fast-propagation algorithm, etc. [15]. There are also second-order algorithms, such as the conjugate gradient method and the method Levenberg-Marquardt, which are significantly faster. It is advisable to choose the Levenberg-Marquardt algorithm as a teaching method. Only with a significant increase in the number of ANN links is the conjugate gradient method used [16].

3.3. Evaluation of ANN adequacy

Based on these steps in the MATLAB environment, we have developed a program. Using the program, we built an ANN model with one hidden layer. We determine the optimal number of neurons in the models by the augmentation method. The search for the optimal number of neurons is performed from 1 to 25.

The criterion for assessing the quality of ANN models is the total maximum absolute error for all nodes - *TMaxAE*. *TMaxAE* is determined for each sample (training, test and control) (Eq.9-11):

$$TMaxAE_{tr} = \sum_{n=1}^{9} max\left|V_i - V_{m,i}\right|, i = 1,2,\dots,trD \tag{9}$$

$$TMaxAE_{ts} = \sum_{n=1}^{9} max\left|V_i - V_{m,i}\right|, i = 1,2,\dots,testD \tag{10}$$

$$TMaxAE_{ch} = \sum_{n=1}^{9} max\left|V_i - V_{m,i}\right|, i = 1,2,\dots,chD \tag{11}$$

where, n - the number of DN nodes except head one; V_i - actual voltage value; $V_{m,i}$ - the voltage obtained by the ANN model; $trD = 1042$, $testD = 446$, $chD = 366$ - the amount of I/O data on the training, test and checking samples, respectively.

Next, a total value is calculated (Eq.12):

$$TMaxAE = TMaxAE_{tr} + TMaxAE_{ts} + TMaxAE_{ch} \tag{12}$$

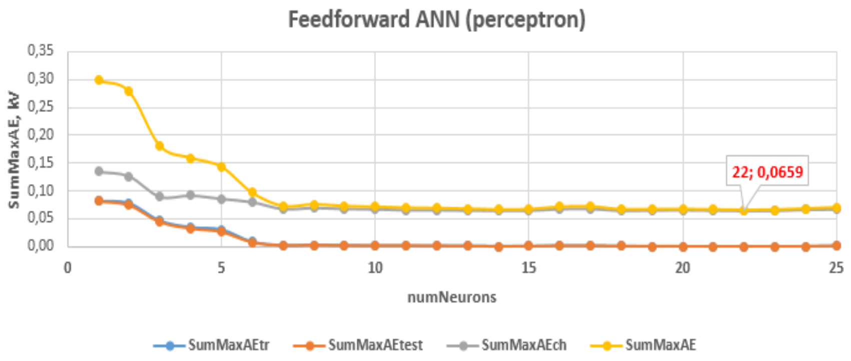

Fig. 2. Optimal number of neurons in the perceptron

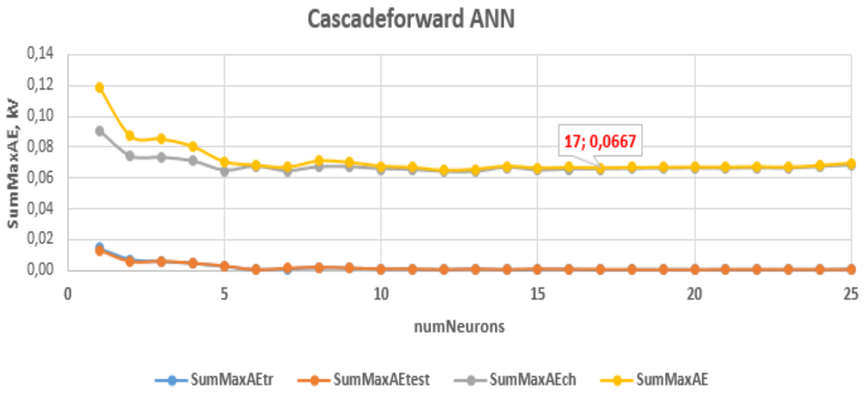

Fig. 3. Optimal number of neurons in the cascade network

So, when using a perceptron, a single-layer ANN with 22 neurons gives the best results (Figure 2). Similar calculations were performed for a cascade forward network where the best results are obtained by a single-layer ANN with 17 neurons (Figure 3).
The results of calculating the *TMaxAE* for the perceptron are presented in Table 1 and for cascade forward network are presented in Table 2.

Table 1. The total maximum absolute error for perceptron

NodeName	unit	Train	Test	Check
U1	kV	0.0000748	0.0000646	0.0081936
U2	kV	0.0000661	0.0000584	0.0110546
U3	kV	0.0000824	0.0000725	0.0126292
U4	kV	0.0000929	0.0000633	0.0137931
U5	kV	0.0000951	0.0000619	0.0144105
U6	kV	0.0000673	0.0000904	0.0011722
U7	kV	0.0000328	0.0000273	0.0012054
U8	kV	0.0000637	0.0000501	0.0011716
U9	kV	0.0000393	0.0000355	0.0012139
TMaxAE	kV	**0.0006145**	**0.0005240**	**0.0648440**

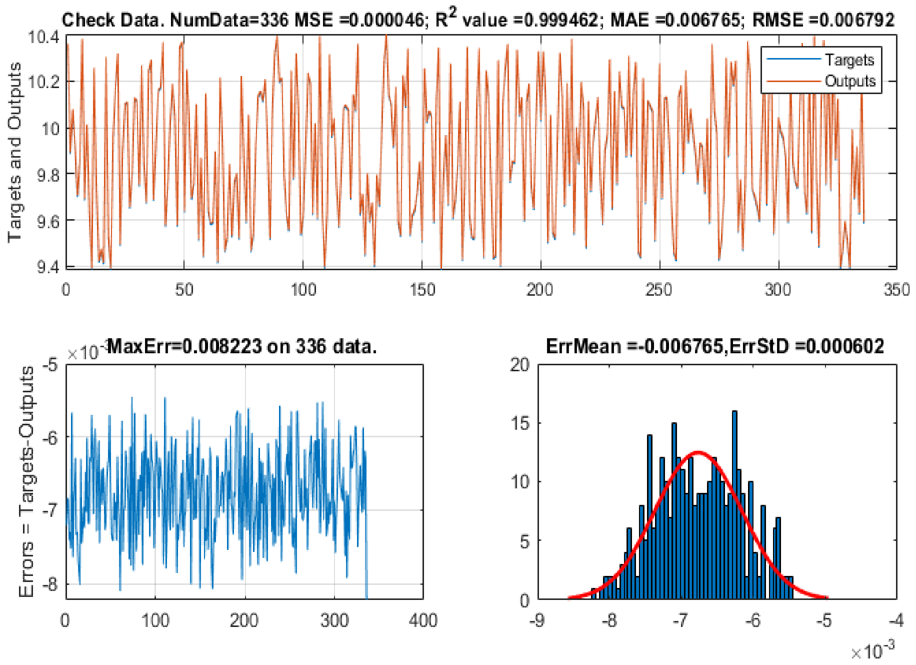

Fig. 4. The results of the assessments on the control sample

In Table 1. the *TMaxAE* on the training sample is 0.0006 kV, on the test sample 0.0005 kV and on the control sample 0.0648 kV. The total value for all samples is 0.0659 kV.
In Table 2, the *TMaxAE* on the training sample is 0.0004 kV, on the test sample 0.0006 kV and on the control sample 0.0657 kV. The total value for all samples is 0.0667 kV.
As can be seen from the results, the single-layer ANN with 22 neurons is the best model for estimating DN node voltages.
As an example, Figure 4 shows the results of node 1voltage estimates on the checking sample.

Table 2. The total maximum absolute error for cascade forward network

NodeName	unit	Train	Test	Check
U1	kV	0.0000392	0.0000481	0.0082227
U2	kV	0.0000335	0.0000328	0.0112106
U3	kV	0.0000366	0.0000388	0.0128244
U4	kV	0.0000391	0.0000705	0.0139288
U5	kV	0.0000589	0.0001661	0.0144178
U6	kV	0.0000599	0.0000625	0.0011670
U7	kV	0.0000600	0.0000633	0.0013812
U8	kV	0.0000452	0.0000461	0.0012715
U9	kV	0.0000628	0.0000537	0.0013023
TMaxAE	kV	**0.0004354**	**0.0005818**	**0.0657263**

Next, we will perform a comparative analysis of the simulation results. To do this, we compare the values of the node voltages according to the ANN models with the results obtained using the backward/forward method. To perform calculations, we use the following initial data (Table 3).

Table 3. Input data for Backward/forward and method ANN model

Input	Ugu	P6	Q6	P7	Q7	P8	Q8	P9	Q9
units	kV	MWt	MVar	MWt	MVar	MWt	MVar	MWt	MVar
Value	10.462	0.08	0.128	0.081	0.124	0.026	0.041	0.026	0.04

The results of the comparative analysis (Table 4) show that the error does not exceed 0.5%. This indicates the possibility of using the obtained ANN for assessing the voltage of nodes in the presented distribution network.

Table 4. The results of the comparative assessment

Output	unit	Target	ANN	Error	Error.%
U1	kV	10.355	10.362	-0.007	0.063
U2	kV	10.322	10.331	-0.010	0.093
U3	kV	10.303	10.314	-0.011	0.108
U4	kV	10.290	10.302	-0.012	0.117
U5	kV	10.283	10.296	-0.013	0.124
U6	kV	0.401	0.402	-0.001	0.322
U7	kV	0.400	0.401	-0.001	0.190
U8	kV	0.400	0.401	-0.001	0.211
U9	kV	0.400	0.401	-0.001	0.218

4. Conclusions

Feedforward artificial neural networks are an alternative (but not a replacement) to traditional methods for on-line determination of the power flow parameters of an open-loop distribution network.

The use of an already trained ANN for calculating the parameters of the steady-state mode requires insignificant computational and time resources in comparison with classical methods, which is very important in operational calculations.

To assess power flow parameters, we obtain the best results when using a single-layer perceptron with 22 neurons. For this ANN, the total maximum absolute error: on the training sample 0.0006 kV, on the test sample 0.0005 kV and on the control sample 0.0648 kV. The total value for all samples is 0.0659 kV.

The results of the comparative analysis show that the error does not exceed 0.5%. This indicates the possibility of using the obtained ANN for assessing the voltage of nodes in the presented distribution network.

References

1. V.I. Idelchik, Electrical systems and networks, Energoatomizdat, Moscow (1989)
2. D. Ortega, V. Reynboldt, Iterative methods for solving nonlinear equation systems with many unknowns, World Publ., Moscow (1975)
3. S.G. Ankaliki, S.G. Gollagi, Power System Steady State Monitoring Using Artificial Neural Network, *Journal of Engineering and Technology* **1**, 4-9 (2011)
4. N, Aishwarya, M.S. Sureban, Analysis of Steady State Stability of Power System using Artificial Neural Network, *International Journal of Scientific & Engineering Research* **10**, 73-77 (2019)
5. M. Khudayarov, N. Normamatov, Power system steady state calculations using artificial neural networks, *E3S Web of Conferences* **216**, 01102 (2020)
6. G.S. Fishman, Monte Carlo: Concepts, Algorithms, and Applications, Springer, New York (1996)
7. R. Hecht-Nielsen, “Kolmogorov's Map: Concepts, Algorithms, and Applications, Springer-Verlag, New York (1987)
8. V.V. Kruglov, V.V. Borisov, Artificial neural networks, Theory and practice 2nd ed, Stereotype, Moscow (2002)
9. A.N. Gorban, D. Wunsch, The general approximation theorem, Proceedings of the IJCNN, IEEE, Anchorage (1998)
10. K.G. Sheela, S.N. Deepa, Review on methods to fix number of hidden neurons in neural networks, *Mathematical Problems in Engineering* **2013**, 1–11 (2013)
11. V. Kurkova, Kolmogorov’s theorem is relevant, *Neural Computation* **3**, 617–622 (1991)
12. Y. Liu, J.A. Starzyk, Z. Zhu, Optimizing number of hidden neurons in neural networks, *Artificial Intelligence and Applications* **2**, 138–143 (2007)

13. H.C. Yuan, F.L. Xiong, X.Y. Huai, A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy, *Computers and Electronics in Agriculture* **40**, 57–64 (2003)
14. A.N. Refenes, E.B. Chan, Sound recognition and optimal neural network design, *Microprocessing and Microprogramming* **35**, 783–789 (1992)
15. D.V. Vasenkov, Methods for training artificial neural networks, *Computer Tools in Education* **1**, 20-29 (2007)
16. S. Osovsky, Neural networks for information processing, Finance and Statistics, Moscow (2002)