A Novel Utilization of NARX for Antenna Array Adaptive Beamforming

Ioannis Mallioras*[(1),(2)], Zaharias D. Zaharis[(1)], Pavlos I. Lazaridis[(3)],
Nikolaos V. Kantartzis[(1)], Traianos V. Yioultsis[(1)], Bo Liu[(4)], Stavros Kalafatis[(5)]
(1) School of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece
(2) Maggioli S.p.A., 47822 Santarcangelo di Romagna, Italy
(3) School of Computing and Engineering, University of Huddersfield, Huddersfield HD1 3DH, U.K.
(4) School of Engineering, University of Glasgow, Glasgow G12 8QQ, U.K.
(5) Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, U.S.A.

Abstract

In this paper, we investigate the use of a nonlinear autoregressive network with exogenous inputs (NARX) for adaptive beamforming on smart antennas. As a beamformer, NARX receives the angles of arrival of incoming signals to extract the complex feeding weights that produce the appropriate antenna radiation pattern. In order to demonstrate the potential of such an implementation, we test our model on a realistic linear antenna array composed of 16 microstrip elements. We use the null steering beamforming technique to produce the datasets needed for training and testing of our model and then we evaluate the accuracy of the radiation patterns produced by this model. To further demonstrate the efficiency of the NARX implementation, we also make a comparison with a feed-forward neural network that has the same architecture with that of NARX.

1. Introduction

Adaptive beamforming (ABF) is a process used by an antenna array to dynamically steer the main lobe of its radiation pattern towards the direction of a desired incoming signal (i.e., signal of interest or SoI), while placing nulls towards the directions of respective interfering signals (i.e., signals of avoidance or SoAs), to finally minimize the signal to interference-plus-noise ratio (SINR). The algorithm responsible for this operation, also known as a beamformer, needs to control the radiation pattern of the antenna array by manipulating the feeding weight of each array element. These weights can purposefully be produced using information about the direction of the incoming signals. The high complexity of conventional deterministic beamformers, such as the minimum variance distortionless response method or the null steering beamforming (NSB) method, makes them inapplicable in an ever-changing environment such as that of 5G and beyond 5G wireless communications. Neural networks (NN) on the other hand, continuously prove themselves to be a fast and reliable alternative to many high-complexity algorithms of different scientific fields [1].

2. NSB algorithm

In order to train any NN for ABF, we need to choose an accurate deterministic ABF technique as a base model. The main advantage of the NSB algorithm is that it can provide great accuracy even in a high-noise environment being very precise at the placement of nulls towards the directions of SoAs. Considering a linear antenna array with M elements and N+1 incoming signals (we identify the first as SoI and the rest N to be SoAs), NSB takes the angles of arrival (AoAs) of the incoming signals as input and calculates the weights needed for the desired radiation pattern. The weight vector produced by this beamformer is calculated using the array steering matrix that corresponds to AoAs of all the incoming signals. When applying the NSB to a realistic linear antenna array composed of M microstrip elements, as shown in [2], the non-isotropic nature of the elements and the coupling between them are both taken into consideration within this steering matrix. Thus, the weights are calculated based on the following expression:

$$\mathbf{w}_{NSB} = \mathbf{A}(\mathbf{A}^H\mathbf{A})^{-1}\mathbf{v}_1, \quad (1)$$

where $\mathbf{A}$ is the $M \times(N+1)$ array steering matrix that corresponds to AoAs of the incoming signals, and $\mathbf{v}_1$ is a unit vector $[1\ 0\ \ldots\ 0]^T$ of size N+1, while superscripts T and H indicates the transpose and the Hermitian transpose operation, respectively. Further analysis and explanation of (1) can be found in [2] and is beyond the scope of this paper.

3. Dataset production

The antenna array of this study consists of 16 elements, where each element is excited with a complex feeding weight. Therefore, 32 weight numbers are needed to produce the radiation pattern (i.e., 16 real and 16 imaginary parts of the feeding weights). To produce such data, some restrictions concerning the desired AoAs and their deviations from the actual AoAs are applied. In particular, AoAs of the incoming signals must lie within the angular sector $[30°, 150°]$ and have a minimum distance of $\Delta\theta$=6°

between each other. Finally, the signal-to-noise ratio (SNR) will be 0 dB, thus considering high noise conditions.

Hence, the produced data set consists of inputs and outputs, where the inputs are the N+1 AoAs with the first one corresponding to SoI and the rest of them to SoAs, while the outputs are the 32 weight numbers created by the NSB algorithm. So, 10^6 records are created for the training of each NN structure, whereas 1.1×10^4 records will be used in the process of searching the best NN architecture. To speed up the process of searching the best architecture for each type of NN, we assume the simple case of 1 SoI and 2 SoAs. AoAs and output weights have been normalized in the range [0,1] to improve the performance of the optimization algorithm.

4. NARX approach

The main difference between this type of NN and a normal feed-forward NN (FFNN) is that it utilizes past values of its output to improve its ability to predict future values. In every training step, it attempts to better approach the desired outcome, not only by relating it to the new input but also to its previous prediction. Another use of such type of NN in the field of wireless communications have been noticed in [3]. In Fig. 1, we show the general structure of a NARX implementation with two hidden layers of size 256 and 512, respectively. The reason why we chose a NARX architecture with two hidden layers that have these specific sizes will be explained below.

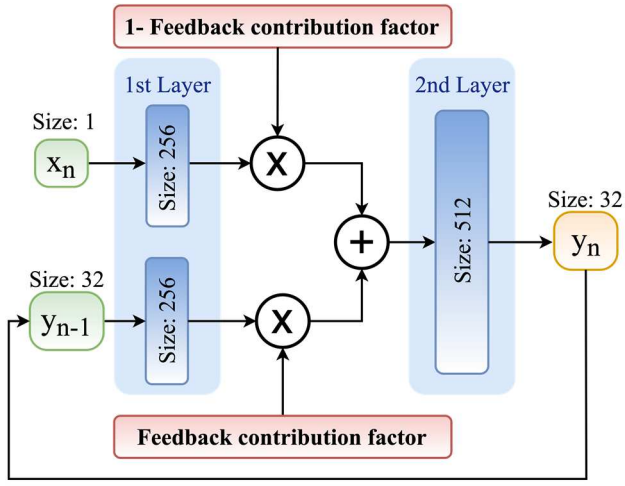

Figure 1. Two-layer NARX implementation.

The incoming AoAs x_n (n = 0,1,..,N) enter the NN in series, and at each step n, the output weight vector y_n is produced using the current input x_n and the feedback of the previous output y_{n-1}. The only input that goes through the first layer without any additional feedback information is, of course, the first one (x_0) that represents SoI (since there is no y_{-1}).

The feedback contribution factor is used to regulate the contribution of the feedback over the actual input in the prediction process and it takes a value within the range [0,1]. To decide about the best setting for this value, we perform grid search using the architecture shown in Fig. 1 with a 3-fold cross-validation to further validate our findings. We use 10^4 records for training and 10^3 records for testing (i.e., 1.1×10^4 records in total), and the root mean square error (RMSE) as a cost function for the training process. From the results shown in Fig. 2, we conclude that the most promising value for the feedback contribution factor is 0.4.

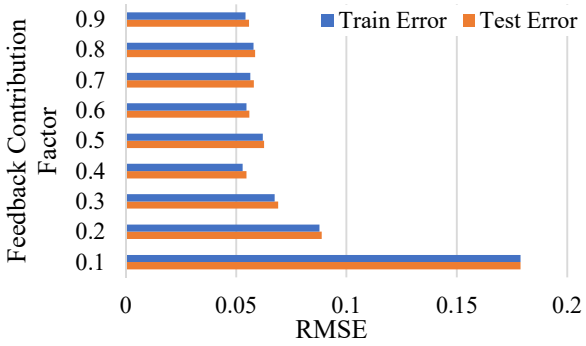

Figure 2. Grid search for feedback contribution factor.

Next, we need to find the best architecture for this type of NN. We test upon a two and a three hidden-layer approach with different sizes for each layer. Each architecture is described by using the notation: [Size of 1st hidden layer, Size of 2nd hidden layer, … , Size of last hidden layer]. The results are given in Fig. 3. It is obvious that the two hidden layer architecture described as [256, 512] provides very satisfactory RMSE value (either during training or during testing), while the use of a third hidden layer only increases the training time seriously without significantly improving the RMSE. Thus, we choose the NARX architecture with two hidden layers of size 256 and 512, respectively, as illustrated in Fig. 1.

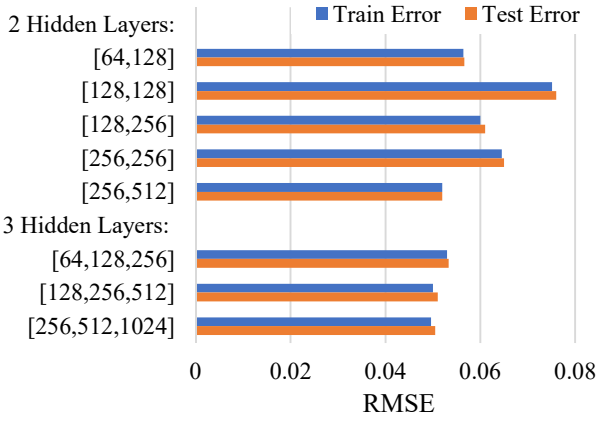

Figure 3. Comparison among various NARX architectures with 2 and 3 hidden layers.

We proceed to train and test the final NARX model using a big dataset of 10^6 records, a batch size of 512, and a learning rate of 0.001. During training, we use the Pytorch

function *ReduceLROnPlateau*, which is set to reduce the learning rate by a factor of 0.8, if the training RMSE stops decreasing or it decreases very slowly. This learning rate regulation increases the training performance by a factor between 2 and 10 [4]. When the learning rate has dropped to a point where it no longer contributes to the training process, or when the test error does not seem to improve over time, we stop the training process. The results are presented in Table I.

Table I. NARX training and test results.

Epochs	Training RMSE	Test RMSE	Final learning rate	Training time (hours)
380	0.0256	0.0257	0.0003	3.5

Then, we test NARX in terms of accuracy in the produced radiation patterns using 10^4 triads of AoAs (every triad corresponds to a SoI and two SoAs) for which NARX has no prior "experience". A statistical analysis of the results derived from this test is given in Table II. It appears that NARX is able to steer the main lobe towards the desired direction (i.e., the direction of SoI) as accurately as the deterministic NSB algorithm. However, NARX faces some difficulty in placing nulls at the desired directions (i.e., the directions of SoAs) as shown by the higher mean value of nulls divergence. This difficulty explains the lower SINR value achieved by NARX compared to the SINR value achieved by the NSB.

Table II. Comparison between NSB and NARX.

	NSB [Mean/Std]	NARX [Mean/Std]
Divergence of main lobe (°)	0.432/0.324	0.411/0.317
Divergence of nulls (°)	0.000/0.000	0.856/0.773
SINR (dB)	27.220/2.982	24.170/4.063

5. Feed-forward NN approach

In the case of FFNNs, the prediction process is much simpler. AoAs (**x** vector shown in Fig. 4) are loaded in the FFNN in parallel and are consequently processed by the neurons of the hidden layers. The NN produces 32 output values (**y** vector shown in Fig. 4).

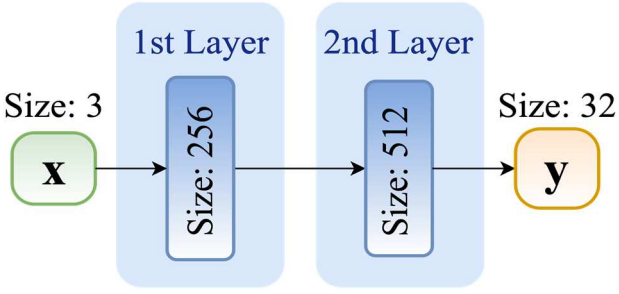

Figure 4. Two-layer FFNN implementation.

The same process as in the case of NARX is followed here, to find the best FFNN architecture. The results of the grid search are presented in Fig. 5. Once again, the use of a third hidden layer does not seem to improve the training and test RMSE enough to justify the increased complexity induced by the extra hidden layer. Thus, we choose the FFNN architecture with two hidden layers of size 256 and 512, respectively, as illustrated in Fig. 4.

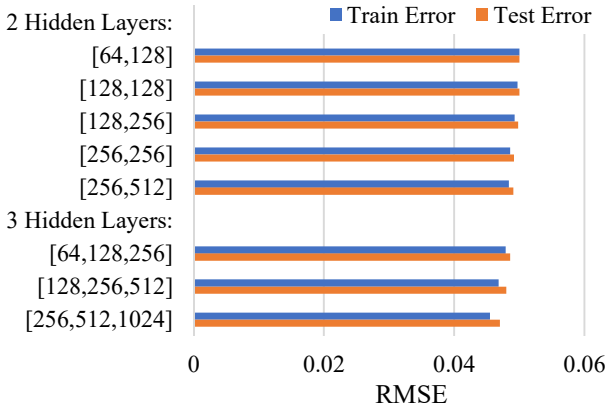

Figure 5. Comparison of FFNNs with 2 and 3 hidden layers.

Next, we train the FFNN model using the same big dataset of 10^6 records and the same hyperparameters we used for the NARX approach. We again stop training when the learning rate drops to the point where it no longer contributes to the training process. The results are given in Table III. Although it seems during grid search (see Fig. 5) that the FFNN model can reach lower levels of RMSE than the NARX model, when we perform longer training the FFNN does not continue to improve like NARX. Despite the fact that the FFNN needs less training time than NARX, it cannot reach the RMSE values achieved by NARX. This also has an impact on the mean value of the divergence of nulls positions, as shown in Table IV. Once again, the statistical analysis given in this Table has been performed by using 10^4 triads of AoAs for which the FFNN has no prior experience.

Table III. FFNN training and test results.

Epochs	Training RMSE	Test RMSE	Final learning rate	Training time (hours)
281	0.0402	0.0415	0.0003	1.7

Table IV. Comparison between NSB and FFNN.

	NSB [Mean/Std]	FFNN [Mean/Std]
Divergence of main lobe (°)	0.432/0.324	0.418/0.319
Divergence of nulls (°)	0.000/0.000	1.722/1.152
SINR (dB)	27.220/2.982	19.540/4.801

6. Comparison between NARX and FFNN

All previous results are summarized in Table V for comparison. The measurements have been performed in the Google Colaboratory environment, using an Intel® Xeon® CPU @2.30GHz with 12GB of RAM (assigned by the Google Colaboratory environment). It is evident that both the FFNN and NARX are much faster than the NSB algorithm (see mean response time in Table V). In addition, Table V shows that the proposed NARX architecture outperforms the respective FFNN architecture in terms of null placement accuracy and SINR level. Of course, the more complex structure of NARX leads to an increased response time compared to that of the FFNN. However, this time remains much shorter than that of the NSB algorithm.

Table V. Comparison between all beamformers.

	Mean divergence of main lobe (°)	Mean divergence of nulls (°)	Mean SINR (dB)	Mean response time (sec)
NSB	0.432	0.000	27.220	1.47
FFNN	0.418	1.722	19.540	0.0002
NARX	0.411	0.856	24.170	0.0010

7. Conclusion

With this study we propose a novel NARX implementation to be used in antenna array adaptive beamforming and we compare it with a FFNN to demonstrate its better performance. We use grid search and the k-fold cross-validation method to find the best settings and the best architectures of NARX and FFNN models, and then we train and evaluate the final versions of these models. The evaluation results have shown that, compared to an FFNN with the same architecture, the proposed NARX is slightly slower in response but more accurate in positioning both the main lobe and the nulls of the radiation pattern.

6. Acknowledgements

This research was supported by the European Union, partially through the Horizon 2020 Marie Skłodowska-Curie Innovative Training Networks Programme "Mobility and Training for beyond 5G Ecosystems (MOTOR5G)" under grant agreement no. 861219, and partially through the Horizon 2020 Marie Skłodowska-Curie Research and Innovation Staff Exchange Programme "Research Collaboration and Mobility for Beyond 5G Future Wireless Networks (RECOMBINE)" under grant agreement no. 872857.

References

[1] D. Erricolo et al., "Machine learning in electromagnetics: A review and some perspectives for future research," *Int. Conf. Electromagnetics in Advanced Applications (ICEAA)*, Granada, Spain, Sep. 2019, pp. 1377–1380, doi: 10.1109/ICEAA.2019.8879110.

[2] Z. D. Zaharis, I. P. Gravas, P. I. Lazaridis, T. V. Yioultsis, C. S. Antonopoulos, and T. D. Xenos, "An effective modification of conventional beamforming methods suitable for realistic linear antenna arrays," *IEEE Trans. Antennas Propag.*, vol. 68, no. 7, pp. 5269–5279, July 2020, doi: 10.1109/TAP.2020.2977822.

[3] P. Bhadauria, R. Kumar, and S. Sharma, "Performance dependency of LSTM and NAR beamformers with respect to sensor array properties in V2I scenario", arXiv, 2021, arXiv:2102.08680.

[4] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *Advances in neural information processing systems*, pp. 2933–2941, June 2014, arXiv:1406.2572.