Lagrange Programming Neural Network for Constraint Antenna Array Beamforming

Bowen Di
Electronics and Information College
Northwestern Polytechnical University
Xi'an, China
E-mail: dbw@mal.nwpu.edu.cn

Huiling Zhao, Zhonghui Zhao, Zhixin Sun, Junjie Tang, Song Lin
Electronics and Information College
Northwestern Polytechnical University
Xi'an, China

***Abstract*— To improve the generalization and real-time anti jamming ability of the beamformer, a novel phase only antenna array beamformer based on Lagrange programming neural network (LPNN) is proposed in this paper. By using the auxiliary variable neurons and Lagrange neurons, the efficiency of LPNN based beamformer improved compared to back propagation neural network (BPNN) one. Training data are collected by the differential evolution algorithm (DE). The simulation results show that the proposed beamformer has a good generalization ability but with a simpler network structure.**

Keywords—Lagrange programming neural network (LPNN), Back propagation neural network (BPNN), beamforming, phased only,

I. INTRODUCTION

Smart antenna is one of the leading innovations in the area of mobile communication and has developed rapidly in recent years[1]. The main aim of the adaptive nulling technique is to steer the mainlobe of the antenna array to the desired direction and place a notch in the interference direction, and ensures the survivability of the adaptive antenna in the complex electromagnetic environment [2].

The common beamforming methods include Least mean square (LMS), Sample matrix inversion (SMI) and Minimum variance distortionless response (MVDR)[3]. With the continuous development of antenna array, the real-time anti-jamming ability is becoming more and more significant. In [4], authors had reviewed and compared different DOA estimation, blind and non-blind ABF algorithms. However, the calculation of adaptive weighting coefficients is computationally intensive, especially in the case of a larger number of antenna array elements. In [5], author had employed the back propagation neural network for a high accuracy channel estimation to simulate a realistic security communication scenario. **In [6], an artificial Feed Forward Neural Network (FFNN) is applied for smart antenna adaptive beamforming. But using neural network only to deal the problem of beamforming may result in the beamformer performs undesirably.**

In this work, the main purpose is to improve the processing time of antenna array nulling by employing neural network, the same to improve the performance of the beamformer. By combining the Lagrange multiplier with the neural network's objective function, establish the Lagrange programming neural network(LPNN) [7] model to realize the real-time antenna array nulling. Based on the method of the Lagrange multipliers, LPNN finds a minimum point of the cost function as well as the solution at an equilibrium point, and leading the dynamic trajectory into feasible region [8]. In order to evaluate the performance of the proposed LPNN, multi-target such as minimum MSE, minimum absolute mean error, lowest failure ratio and the distribution of the null level are considered. By comparing back propagation neural network and Lagrange programming neural network's simulation results, we find that beamformer based on LPNN shows a considerable improvement in generalization ability.

II. BEAMFORMING MODEL

A. Training set preparation

In this work, constrained differential evolution algorithm (DE)[9] is used to set up the constraint model. A N-elements uniform linear array pattern can be written as:

$$AF(\theta) = \sum a_n e^{j(kd\cos(\theta_n)+\varphi_n)} \tag{1}$$

Where the a_n is the amplitude excitation, φ_n is the phase excitation of the n th element respectively and d is the distance between the elements. Linear array's sidelobe level can be written as:

$$SLL(\theta) = 20log_{10}\left|\frac{AF\ (\theta)}{max\ (AF)}\right|_{\theta\in\Theta_k} \tag{2}$$

Where Θ_k denote the sidelobe. In order to form null at the interference direction, the weights of elements will be adjusted. However, forming nulls may lead to the increasing of sidelobe levels and the loss of the power on the mainlobe. Therefore, a constraint model based on beamforming problem, as is shown in formula (3), is structured to solve this problem.

$$\begin{aligned} &min && max(SLL) && \\ &subject\ to && max(SLL_{null}) - \delta n \le 0 && (a) \\ & && max((SLL_{near}) - \delta s \le 0 && (b) \\ & && AF(\theta_m) - max(AF) = 0 && (c) \\ & && \delta m - AF(\theta_m) \le 0 && (d) \end{aligned} \tag{3}$$

Where constraint (a)、(b) define the allowed maximum level of nulls(δ_n) and the near sidelobe level(δ_s), Constraint (c) forces the array highest peak steering at desired direction which is of great importance in beamforming. Constrain (d) forces the level of the highest peak no less than δ_m.

B. Bcak propagation neural network

Back propagation neural network [10] [11] is based on error back propagation. The basic idea of the BPNN is to calculate the error via comparing the sample's outputs with the network's outputs and take this as a reference to adjust the nulls' weights and biases.

Assuming the BPNN is three-layer structure, and the sample size is p. The weight between ith input layer node and kth hidden layer node can be expressed as v_{ki}, and w_{jk} represent the weight between kth hidden layer node and jth output layer node. So, the kth hidden layer node output can be written as:

978-1-5386-7302-7/18/$31.00 ©2018 IEEE

$$z_{pk} = f(net_{pk}) = f(\sum_{i=0}^{n} v_{ki} x_{pi}) \tag{4}$$

The *j*th output layer node output can be shown as:

$$y_{pj} = f(net_{pj}) = f(\sum_{k=0}^{q} w_{jk} z_{pk}) \tag{5}$$

The global error function can be defined as :

$$E = \sum_{p=1}^{p} E_p = \frac{1}{2} \sum_{p=1}^{p} \sum_{j=1}^{m} (t_{pj} - y_{pi})^2 \tag{6}$$

Where E_p is pth sample error, and t_{pj} is the training set output.

The adjusting formula of the output layer weights is

$$\Delta w_{jk} == \eta \sum_{p=1}^{p} (t_{pj} - y_{pi}) \cdot y_{pj}(1 - y_{pj}) \tag{7}$$

The adjusting formula of the hidden layer weights is:

$$\Delta v_{ki} = \eta \sum_{p=1}^{p} (\sum_{j=1}^{m} (\delta_{pi} \cdot w_{jk}) \cdot z_{pk}(1 - z_{pk}) \cdot x_{pi} \tag{8}$$

C. Lagrange programming neural network

The Lagrange programming neural network [12] ,based on Lagrange multiplier method, does well in solving general nonlinear constrained optimization problems. Instead of considering them in a digital way, we construct an analog neural network and define the neural dynamics to govern the state transition of the neurons. After the network settles down at one of equilibrium points, the solution is obtained by measuring the neuron outputs at this stable equilibrium point.

Consider a general nonlinear constrained optimization problem, given by

$$\begin{cases} min \quad f(x) \\ s.t. \ h(x) = 0 \end{cases} \tag{9}$$

Where $\boldsymbol{x}$=[x_1 x_2 .... x_J] is the variable vector, $f(x)$ is the objective function, and $h(x)$ is a vector valued function that describes the equality constraints.

The LPNN approach first constructs a Lagrangian function, shown as the following:

$$min \ L(x, \lambda) = f(x) + \lambda^T h(x) \tag{10}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a vector containing the Lagrange multipliers. In order to achieve the optimum solution of the formula (8), we can deduce the transient behaviors:

$$\begin{cases} \frac{\partial L(x,\lambda)}{\partial x} = \frac{\partial f(x)}{\partial x} + \lambda^T \frac{\partial h(x)}{\partial x} = 0 \\ \frac{\partial L(x,\lambda)}{\partial \lambda} = h(x) = 0 \end{cases} \tag{11}$$

In order to hold x and λ, we set up two kinds of neurons, the variable neurons and the Lagrange neurons, which work simultaneously and their dynamics are shown as following:

$$\begin{cases} \frac{dx}{dt} = -\frac{\partial L(x,\lambda)}{\partial x} = -\frac{\partial f(x)}{\partial x} - \lambda^T \frac{\partial h(x)}{\partial x} \\ \frac{d\lambda}{dt} = \frac{\partial L(x,\lambda)}{\partial \lambda} = h(x) \end{cases} \tag{12}$$

D. LPNN Model

With the aid of augmented Lagrangian function, and combining with the global error function of BPNN, we recast the beamforming problem as a constrained optimization problem, and combining with the global error function of BPNN,

$$\begin{cases} min \quad f(x) = \frac{1}{2} \sum_{p=1}^{p} \sum_{j=1}^{m} (t_{pj} - y_{pi})^2 \\ s.t. \quad h(x) = SLL_{null} - \delta n = 0 \end{cases} \tag{13}$$

Where SLL_{null} is the nulls level and δn is the allowed maximum level of nulls. The Lagrangian function is written as:

$$\begin{aligned} min \ L(x, \lambda) &= f(x) + \lambda^T h(x) \\ &= \frac{1}{2} \sum_{p=1}^{p} \sum_{j=1}^{m} (t_{pj} - y_{pi})^2 \\ &\quad + \lambda^T (SLL_{null} - \delta n) \end{aligned} \tag{14}$$

As the LPNN is an analog solution, we employ discretisation in our digital implementation:

$$\begin{cases} x(Iter + 1) = x(Iter) + \rho \frac{dx}{dt} \\ \lambda(Iter + 1) = \lambda(Iter) + \rho \frac{d\lambda}{dt} \end{cases} \tag{15}$$

Where $Iter$ means the number of iterations and ρ means the change rate. The continuous-time realization of LPNN is depicted in Fig1, where the integration operators are implemented in parallel and synchronously. When both the Lagrangian and variable neurons arrive at the equilibrium point, the outputs of neural network are the original optimization

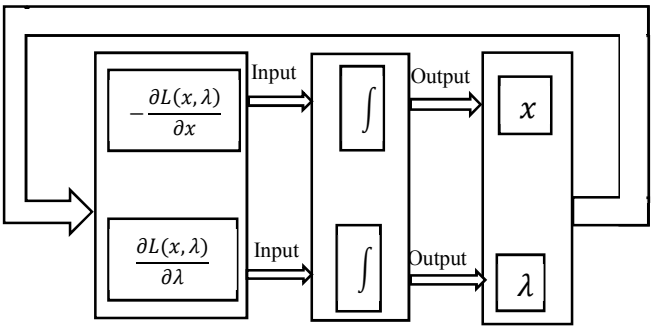

Fig1 The realization of LPNN

In this neural network model, the $\{x, \lambda\}$ represent the state of the neurons. The neurons individually compute the state transition, that is, $-\partial L(x, \lambda)/\partial x$ and $\partial L(x, \lambda)/\partial \lambda$.

The LPNN can be realized by the following steps:

Step1: Collect the relevantly needed data for forecasting and preprocess the raw data. Normalize for the processed data and select the load as output of the LPNN, interference direction and mainlobe direction as input variables of the LPNN.

Step2: Set up the layers of the NN, and the number of the neurons in each layer, select the global error function $f(x)$ and the acceptable error. Initialize the neuron weights w_{ij} and biases b_i.

Step3: Calculate output of each hidden and output layers and update w_{ij} and b_i ,calculate record the output error function E .

Step4: Take the reference of E to calculate Δw_{ij} and Δb_i for the hidden and output layers and update the weights and biases.

Step5: Compute the overall error of all the samples. When the error meets the error request, output the BPNN model, else, go to step3 and start new round iteration.

III. SIMULATION AND ANALYSIS

In this section, BPNN and LPNN are realized to evaluate the performance of the two neural networks for constraint antenna array beamforming. In this study, a 9-elements uniform linear array spaced 0.5λ apart is considered. The initial amplitude of the array has -15dB Chebyshev taper

In this work, the preprocessing training set can be divided to two parts. First, about the input of NN, $\boldsymbol{x} = [x_1\ x_2]$, we employ the normalization method. Then, for the output of the NN, with the aid of electromagnetic knowledge we find that in the view of uniform linear array , the same phase difference will shape the same pattern despite the different y_1. In order to eliminate the different phase combinations corresponding to the same patterns caused by the different y_1,by subtracting $[y_2\ \ y_9]$ to y_1 , transform the $y = [y_1\ y_2\ ....\ y_9]$ into $y = [0, (y_2 - y_1)\ ....\ (y_9 - y_1)]$ and remove contradictory items to improve the sample set's quality.

A. Simulation of BPNN

The BPNN is a four-layer neural network consisting of one input layer, two hidden layers and one output layer. The input vectors of the BPNN is the null direction and the array highest peak direction, denoted as $\boldsymbol{x} = [x_1\ x_2]$, and the outputs are the array elements excitation phases, denoted as $y = [y_1\ y_2\ ....\ y_9]$. The structure of the BPNN is [2,11,5,9], and the learning rate $\varepsilon = 0.03$, the number of iterations=1000 are employed. To start the iterative procedure, the estimates weights and biases are initialized as uniform numbers distributed between 0 and 1.

As the figure 2 shown, the BPNN converges at about 200 iterations and the best MSE is about 0.0076. The absolute mean error is 0.067rad, about 3.4 . According to formula (1) ,the $\Delta\theta = 3.4$ will lead to 0.042 deviation for the exponential term and the deviation is tolerated by the form of the antenna pattern.

The test result shows that the well trained BPNN can place a -30dB notch in the interference direction. The accuracy of the test sample is about 86.6%. It is worth mentioning that by employing BPNN model, the operation speed of constraint linear array antenna beamforming is improved obviously.

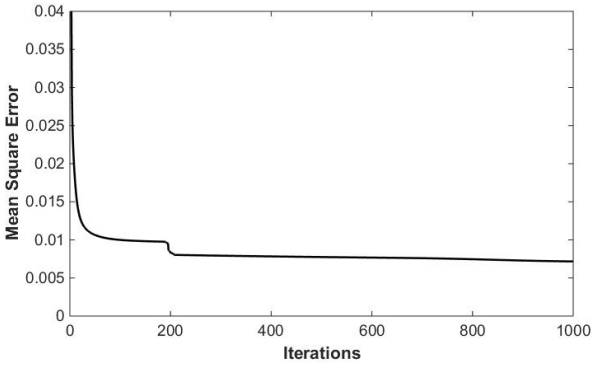

Fig 2 BPNN's dynamic behaviors for Mean square error versus iteration number

B. Simulation of LPNN

The Lagrange programming neural network (LPNN) combines the Lagrange multiplier with the artificial neural network. In this work, Lagrange multiplier is applied to programe the BPNN's objective function, aimed at improving the BPNN model and forming nulls more effectively. The LPNN is a three-layer neural network and the structure is [2,10,9]. The Lagrange multiplier is updated like the formula, $\lambda(Iter+1) = \lambda(Iter) + \rho\frac{d\lambda}{dt}$,and the renew modulus $\rho = 0.05$, the number of the iterations $Iter = 200$.

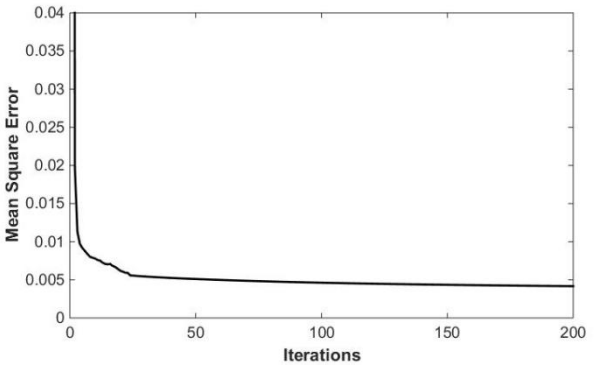

Fig 3 LPNN's dynamic behaviors for Mean square error versus iteration number

From Figure 3 we can find that the LPNN converges before 25 iterations ,compared with the BPNN, the convergence speed is accelerate obviously. The best MSE is about 0.0042, and the best absolute mean error is 0.053. The detail compassions between LPNN and BPNN is in the table 1.

TABLE I. THE COMPARISON OF RESULTS

	Comparison		
	BPNN	***LPNN***	***Improvment***
Mean Square Error	0.0076	0.0042	44.7%
Absolute Mean Error	0.067	0.053	20.9%
Converged Iteration	200	20	90%

	Comparison		
	BPNN	***LPNN***	***Improvment***
Accuracy of the test	86%	94%	8%

By testing the well trained neural network, we summarize that most of substandard samples perform worse in forming ideal nulls level. So we analyze the distribution of the nulls level formed by BPNN and LPNN.

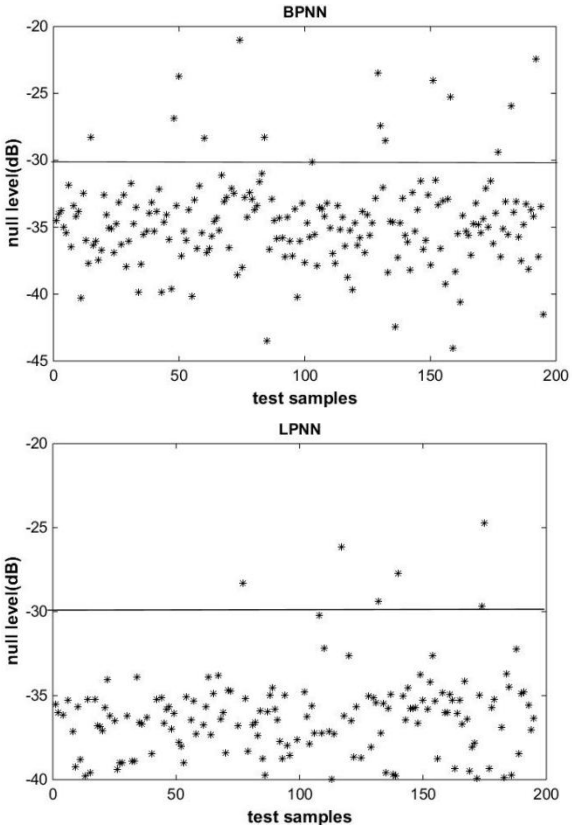

Fig 4 The distribution of the nulls level formed by BPNN and LPNN

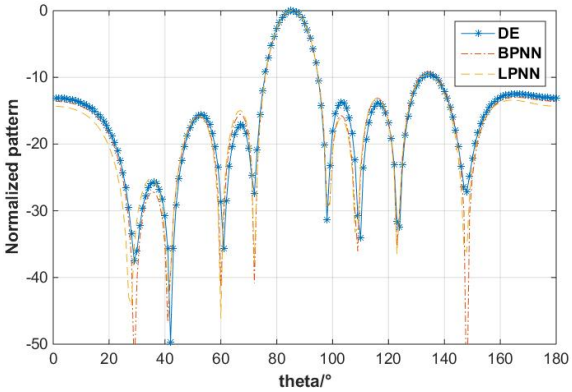

Fig 5 The normalized pattern based on DE, BPNN and LPNN

As is shown in the Fig 4 ,after employ the Lagrange multiplier and add the null constraint conditions to the objective function, the null level predicted by the LPNN shift down obviously compared to BPNN. Most of nulls predicted by BPNN distribute between -30dB and -40dB, and the LPNN distribute between -35dB and -40dB. In addition ,the nulls predicted by LPNN is more concentrated than BPNN. The number of unsatisfied samples is only 5 (the total test sample number is 200), while BPNN is 14. It is worth to say that the BPNN employs four-layer structure and more hidden layer nodes, and LPNN is three-layer structure and uses the simpler structure to achieve a better result.

Fig 5 is the beamforming based on DE,BPNN and LPNN. The mainlobe direction is 85.5° and the inference direction is 42°. It can be seen that all of these methods can successfully steer null in the interference direction and the patterns based on BPNN and LPNN show similar tendency with DE, which proves the good generalization ability of the networks. By the way, LPNN performs better via steering a deeper null than BPNN.

IV. Conclusion

The beamformer designed in this work modifies the phase excitations of the weights and the amplitudes are held constant with -15dB Chebyshev array weight. The simulation results show that the LPNN employs simpler structure to achieve good generalization ability and more accurate. This behavior combined with the advantage of instant response makes the NN-based beamformer very useful in practice.

Acknowledgment

This work is supported by the aeronautical science foundation of China under grant 201420530.

References

[1] Salama S A, Abdalla A Z. Study of smart antenna characteristics using genetic algorithms (GA) applications[C]. Telecommunications Forum. 2013:268-271.

[2] Haupt R L. Phase-only adaptive nulling with a genetic algorithm[J]. IEEE Transactions on Antennas & Propagation, 1997, 45(6):1009-1015.

[3] WENPIN LIAO. Array pattern synthesis with null steering using genetic algorithms by controlling only the current amplitudes[J]. International Journal of Electronics, 1999, 86(4):445-457.

[4] Sharma A, Mathur S. Performance Analysis of Adaptive Array Signal Processing Algorithms[J]. Iete Technical Review, 2016, 33(5):472-491.

[5] Song H, Wen H, Hu L, et al. Cooperative Secure Transmission with Imperfect Channel State Information Based on BPNN[J]. IEEE Transactions on Vehicular Technology, 2018, PP(99):1-1.

[6] Sallomi A H, Ahmed S. Multi-layer feed forward neural network application in adaptive beamforming of smart antenna system[C]// Al-Sadeq International Conference on Multidisciplinary in It and Communication Science and Applications. IEEE, 2016:1-6.

[7] Wang H, Feng R, Leung A C S, et al. Lagrange Programming Neural Network Approaches for Robust Time-of-Arrival Localization[J]. Cognitive Computation, 2017(2):1-12.

[8] Wang H, Leung C S, So H C, et al. Robust MIMO Radar Target Localization based on Lagrange Programming Neural Network[J]. 2018.

[9] Baraldi P, Bonfanti G, Zio E. Differential evolution-based multi-objective optimization for the definition of a health indicator for fault diagnostics and prognostics[J]. Mechanical Systems & Signal Processing, 2018, 102:382-400.

[10] Guo X, Liebgott H, Friboulet D. Back-propagation beamformer design for motion estimation in echocardiography.[J]. Ultrason Imaging, 2015, 37(3):179-204.

[11] Wang B, Jalil N A A, Voon W S. Prediction of rotary machinery degradation status based on vibration data using back propagation (BP) neural network[J]. 2012.

[12] Feng R, Leung C S, Constantinides A G, et al. Lagrange Programming Neural Network for Nondifferentiable Optimization Problems in Sparse Approximation[J]. IEEE Transactions on Neural Networks & Learning Systems, 2016, PP(99):1-1