JOURNAL OF COMPUTING, VOLUME 3, ISSUE 3, MARCH 2011, ISSN 2151-9617
HTTPS://SITES.GOOGLE.COM/SITE/JOURNALOFCOMPUTING/
WWW.JOURNALOFCOMPUTING.ORG

Improved Nelder Mead's Simplex Method and Applications

Nam Pham†, Bogdan M. Wilamowski†
†Electrical and Computer Engineering, Auburn University, Alabama, US

***Abstract* – Nelder Mead's simplex method is known as a fast and widely used algorithm in local minimum optimization. However, this algorithm by itself does not have enough capability to optimize large scale problems or train neural networks. This paper will present a solution to improve this deficiency of Nelder Mead's simplex algorithm by incorporating with a quasi gradient method. This method approximates gradients of a function in the vicinity of a simplex by using numerical methods without calculating derivatives and it is much simpler than analytical gradient methods in mathematic perspectives. With this solution, the improved algorithm can converge much faster with higher success rate and still maintain the simplicity of simplex method. Testing results with several benchmark optimization problems of this improved algorithm will be compared with Nelder Mead's simplex method. Then this algorithm will be applied in synthesizing lossy ladder filters and training neural networks to control robot arm kinematics. These typical applications are used to show the ability of the improved algorithm to solve the varieties of engineering problems.**

***Index Terms* — Nelder Mead's simplex method, quasi gradient method, lossy filter, Error Back Propagation, Lavenberg Marquardt, neural networks**

1. INTRODUCTION

Nelder Mead's simplex method is a direct search method [1]. Its computational process is simple and it does not require calculation of derivatives [2]. However, Nelder Mead's simplex method does not rely on the gradient so it may converge slowly or may not converge at all. This scenario usually happens and remarkably reduces efficiency of Nelder Mead's simplex method in solving complicated problems such as optimizing multi-dimensional cases or training neural networks. To improve its performance, Nelder Mead's simplex method can be incorporated with other techniques such as a quasi gradient method in this paper. The improved simplex method does not require complex mathematic computations can optimize complex multidimensional problems with higher success rate and faster convergence speed. The improved success rate of Nelder Mead's simplex method shows its potential capability in many real applications [3], [4].

In recent years, artificial neural networks (ANNs) have been applied widely in industry as control system, VSLI, medical diagnosis, etc [5], [6], [7], [8], [9]. Although ANNs are very powerful in many applications, but at the same time it is not easy to train neural networks. Many training algorithms are introduced so far but none of them can train for all neural networks. The first order method Error Back Propagation (EBP) can be used to train simple ANNs but with more complex or large ANNs this method is not efficient because of its slow convergence [10]. The second order methods as Levenberg Marquardt (LM) [11], [12] or Neuron by Neuron (NBN) introduced recently [13] can train ANNs 1000 times faster than EBP algorithm. Even though algorithms using Jacobian or Hessian matrix computation [10], [11], [12], [13] converge faster than Error Back Propagation or any algorithm based on steepest descent, they are not suitable for training large networks because their computing time grows proportional to the problem size. There are many algorithms developed to improve Error Back Propagation while maintaining its simplicity [14], [15], [16] but their convergence speed is still slow in order to be more realistic and applicable. Therefore, it is necessary to develop a reliable algorithm can train neural networks without expensive computational cost. Nelder Mead's simplex method with its simple computations does not have to calculate first derivatives, second derivatives or does not have to invert Jacobian or Hessian matrix seems to be a potential algorithm to solve this problem. Unfortunately, Nelder Mead's simplex method does not really have a good success rate and does not converge really well. However, by incorporating a quasi gradient method with Nelder Mead's simplex method, the new algorithm can converge much faster and has ability to train neural networks for the purpose of many practical applications which are nontrivial or impossible for other well known algorithms as Lavenberg Marquardt or Error Back Propagation [17].

This paper is organized as follows. Section 2 reviews Nelder Mead's simplex method. Section 3 presents the improved simplex method by combining with a quasi gradient method. Section 4 presents experimental results. Section 5 presents applications of the improved simplex method in synthesizing lossy filters and training neural networks to control robot arm kinematics. Section 6 is a conclusion.

2. Overview of Nelder Mead's Simplex Method

Despite its age, Nelder Mead's simplex method (SIM) is still a method of choice for many practitioners in the fields of statistics, engineering, and the physical and medical sciences because it is easy to code and very easy to use [18]. It is a fast algorithm to search for a local minimum and applicable for multi-dimensional optimization. It does not have to calculate derivatives to move along a function as gradient methods. It converges to minima by forming a simplex and using this simplex to search for its promising directions. A simplex is defined as a geometrical figure which is formed by (N+1) vertices (N: the number of variables of a function). In each iteration, SIM always starts calculating a reflected point of the worst point through the centroid point. According to this value, SIM algorithm will do reflection or extension, contraction or shrink to form a new simplex. In other words, the function values at each vertex will be evaluated in each iteration and the worst vertex with the highest value will be replaced by another vertex which has just been found. Otherwise, a simplex will be shrunk around the best vertex. This process will be repeated iteratively until a desired error value is satisfied.

Convergence speed of the simplex method may be affected by three parameters α, β, γ (α is the reflection coefficient to define how far the reflected point should be from the centroid point, β is the contraction coefficient to define how far the contracted points should be when they are contracted from the worst point and the reflected point in case the function values at these points are the same, γ is the expansion coefficient to define how far to expand from the reflected point in case a simplex moves to the right direction). Depending on these coefficients α, β, γ, volume of the simplex will be changed by the operations of reflection, contraction or expansion respectively [1]. All steps of Nelder Mead's simplex method can be summarized as following:

```
• Step 1: get α, β, γ, select an initial simplex with random vertices $x_0, x_1,..., x_n$ and calculate their function values.
• Step 2: sort the vertices $x_0.,x_1,..., x_n$ of the current simplex so that $f_0, f_1,..., f_n$ in the ascending order.
• Step 3: calculate the reflected point $x_r, f_r$
• Step 4: if $f_r < f_0$:
    (a) calculate the extended point $x_e, f_e$
    (b) if $f_e < f_0$ , replace the worst point by the extended point $x_n = x_e, f_n = f_e$
    (c) if $f_e > f_0$ , replace the worst point by the reflected point $x_n = x_r, f_n = f_r$
• Step 5: if $f_r > f_0$:
    (a) if $f_r < f_i$, replace the worst point by the reflected point $x_n = x_r, f_n = f_r$
    (b) if $f_r > f_i$:
        (b1) if $f_r > f_n$: calculate the contracted point $x_c, f_c$
            (c1) if $f_c > f_n$ then shrink the simplex
            (c2) if $f_c < f_n$ then replace the worst point by the contracted point $x_n = x_c, f_n = f_c$
        (b2) if $f_r < f_n$: replace the worst point by the reflected point $x_n = x_r, f_n = f_r$
• Step 6: if the stopping conditions are not satisfied, the algorithm will continue at step 2
```

3. Improved Simplex Method with Quasi Gradient

Nelder Mead's simplex method is considered as a fast and simple algorithm. However, its poor convergence restricts its application in class of problems with two or three variables. When optimizing high dimensional problems, Nelder Mead's simplex method can fail to converge easily. Because of this deficiency, this algorithm needs to be altered in some how to be more robust and reliable. For this purpose, many authors propose different ideas to improve it. Fuchang Gao and Lixing Han propose an implementation of the Nelder-Mead method in which the expansion, contraction, and shrinking parameters depend on the dimension of the optimization problem [19]. Another author as Torczon suggests that this poor convergence may be due to the search direction becomes increasingly orthogonal to the steepest descent direction [20], etc. Without any satisfactory convergence theory, but it is clear that the effect of dimensionality should be extended and researched more. This paper is another effort to improve the simplex algorithm which is different from other explanations in the literature. This improved algorithm is addressed by its simplicity which is one of the key factors to make Nelder Mead simplex method so popular.

The major drawback of Nelder Mead's simplex method is that it may not define its moving directions well enough just by simple geometrical movements in high dimensional cases. This explains why Nelder Mead's simplex method is a simple and fast algorithm but is not stable in optimizing multi-dimensional problems.

To illustrate this reasoning we can consider two extreme cases where Nelder Mead's simplex method may not converge to local minima (using 2-d cases for easy illustration). These two cases with the locations of *B* (best), *G* (good), *W* (worst) points have significantly different gradient directions. In the case (a) Fig. 1the function values at *W* and *G* are similar while in the case (b) Fig.2 the function values at *B* and *G* are similar. In both cases, the gradients head to different directions from Nelder Mead's simplex method. When it fails to search in R direction, it will repeat to search in that direction again R_1 which is not the right direction to local minimia. In order to improve speed and convergence rate of the simplex method, it needs to rely on the gradient. With a new way to calculate the reflected point according to the quasi gradient method, a new simplex *ΔBGR'* is created instead of *ΔBGR* Fig. 1, 2.

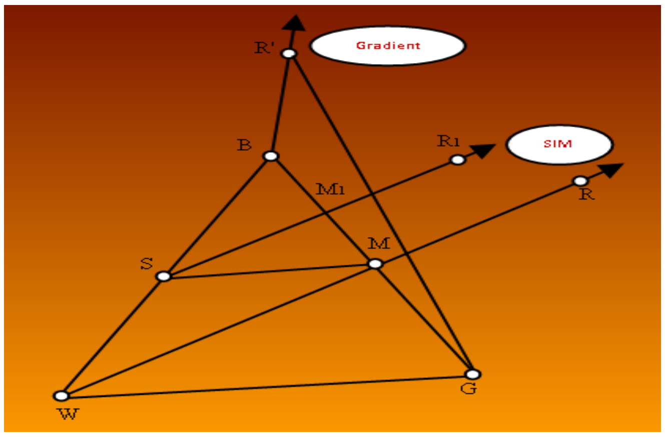

Fig. 1: The triangular simplex *ΔBGW* with similar function values at *W* and *G*

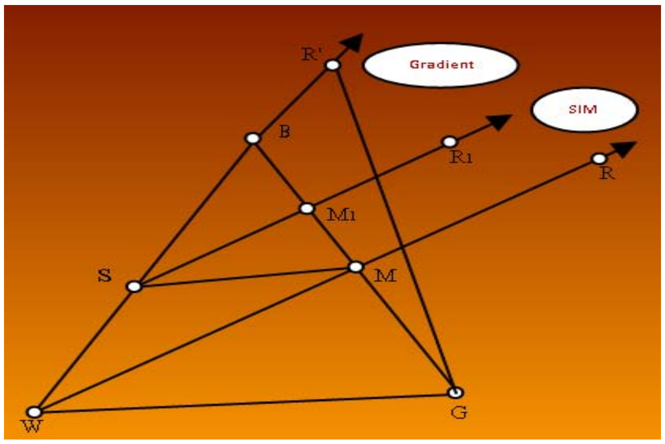

Fig. 2: The triangular simplex *ΔBGW* with similar function values at *B* and *G*

Nelder Mead's simplex method is simple and can converge to local minima without calculating derivatives. To maintain this simplicity, one quasi gradient method is presented to approximate gradients of a function [16]. This method uses an extra point created from a simplex to approximate gradients. The accuracy of this method depends on the linearity of a function in the vicinity of a simplex. However, its computing cost does not increase significantly when the size of optimized problems becomes larger.

This method approximates gradients of a (n+1) dimensional plane created from a geometrical simplex. By approximating gradients of the plane, we can approximate gradients in the vicinity of a simplex. First we select an extra point with its coordinates composed from (n+1) vertices in a simplex and then combine this point with n selected vertices in the same simplex to estimate gradients. Its steps are presented as following:

Assume an optimized function *f*: $P^n \rightarrow P$, $x \in P^n$

- Step 1: initialize a simplex with (n+1) random vertices $x_1, x_2, \ldots, x_n$
- Step 2: select an extra point *xs* with its coordinates composed from n vertices in the simplex. In other words, coordinates of the selected point are the diagonal of matrix *X* from n vertices in the simplex.

$$xs = diag\begin{bmatrix} x_{1,1}\ x_{1,2} \ldots x_{1,n-1}\ x_{1,n} \\ x_{2,1}\ x_{2,2} \ldots x_{2,n-1}\ x_{2,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_{n,1}\ x_{n,2} \ldots x_{n,n-1}\ x_{n,n} \end{bmatrix}_{nxn} \qquad (1)$$

Or $xs = [x_{1,1}, x_{2,2}, \ldots, x_{n,n}]$ (2)

- Step2: calculate quasi gradients based on the selected point *xs* and other n points in the simplex.

```
For i =1: n,
    If mod (i, 2) == 0
        $g_i = \frac{\partial f}{\partial x_i} = \frac{f(i-1) - f(xs)}{x_{i-1,i} - xs_i}$
    Else                                                        (3)
        $g_i = \frac{\partial f}{\partial x_i} = \frac{f(i+1) - f(xs)}{x_{i+1,i} - xs_i}$
    End
End
```

- Step 3: calculating the new reflected point *R'* based on the best point *B* and the approximate gradients. Parameter σ is the learning constant or step size.

$$R' = B - \sigma * G \qquad (4)$$

- Step 4: if the function value at *R'* is smaller than the function value at *B*, it means that *BR'* is the right direction of the gradient then *R'* can be expanded to *E'*.

$$E' = (1-\gamma)B + \gamma R' \qquad (5)$$

The quasi gradient method using numerical methods have just been presented above is much simpler than analytical gradient methods. This method does not have to derive derivatives of a function which is usually very difficult for complicated functions. Generally, the improved simplex method with the quasi gradient method is similar to Nelder Mead' simplex method except the way it calculates the reflected point and the extended point in case the Nelder Mead's simplex method cannot define its moving directions.

4. Experimental Results

All algorithms are written in Matlab and all experiments are tested on a PC with Intel Quad. Several benchmark functions which are well-known in local minimum optimization are tested in these experiments [1], [17], [18], [21]. Each function exhibits features deemed relevant for the purpose of this comparison. In order to compare performances of these two algorithms, some assumptions are set: algorithms start with a random initial simplex in the range of [-100, 100]; dimensions of all benchmark problems are equal to 10, 15, 20 respectively; maximum iteration is equal to 50,000; desired error is predefined to terminate algorithms DE= 0.001; coefficients α= 1, β= 0.5, γ= 2; learning constant σ=1. All results in Table 1-3 are the average values calculated over 100 random running times.

(1) De Jong function 1

$$F_1 = \sum_{i=1}^{n} x_i^2$$

(2) Step function

$$F_2 = \sum_{i=1}^{n} |x_i + 0.5|^2$$

(3) Wood function

$$F_3 = \sum_{i=1}^{n} [\, 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 + 90(x_{i+3} - x_{i+2}^2)^2 + (1 - x_{i+2})^2 + 10.1((1 - x_{i+1})^2 + (1 - x_{i+3})^2) + 19.8(1 - x_{i+1})(1 - x_{i+3})]$$

(4) Powell function

$$F_4 = \sum_{i=1}^{n} [\, (x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + (x_{i+1} - 2x_{i+2})^4 + 10(x_i - x_{i+3})^4]$$

(5) Rosenbrock function

$$F_5 = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

(6) Schwefel function

$$F_6 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$$

(7) Zarakov function

$$F_7 = \sum_{i=1}^{n} [x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i \right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i \right)^4]$$

(8) Biggs Exp6 function

$$F_8 = \sum_{i=1}^{n} [x_{i+2} e^{t_i x_i} - x_{i+3} e^{-t_i x_{i+1}} + x_{i+5} e^{t_i x_{i+4}} - y_i]^2$$

$$where: t_i = 0.5i$$

$$y_i = e^{-t_i} - 5e^{-10t_i} + 3e^{-4t_i}$$

(9) Colville function

$$F_9 = \sum_{i=1}^{n} [\, 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 + (x_{i+2} - 1)^2 + 10.1((x_{i+1} - 1)^2 + (x_{i+3} - 1)^2) + 90(x_{i+2}^2 - x_{i+3})^2 + 19.8(x_{i+1} - 1)(x_{i+3} - 1)]$$

(10) Box function

$$F_{10} = \sum_{i=1}^{n} \left[e^{-ax_i} - e^{-ax_{i+1}} - x_{i+2}(e^{-a} - e^{-10a}) \right]^2$$

$$where: a = t * i, t \text{ is a const}$$

EXPERIMENTAL RESULTS

Function	Nelder Mead's simplex method			Improved simplex method with quasi gradient		
	Success rate	Iteration	Computing time	Success rate	Iteration	Computing time
F_1	100%	989	0.0907	100%	531	0.0806
F_2	100%	935	0.0863	100%	535	0.0825
F_3	41%	2148	0.2042	52%	1259	0.2038
F_4	100%	1041	0.1037	100%	777	0.1346
F_5	55%	9606	0.8812	76%	7993	1.2094
F_6	100%	1772	0.1666	100%	898	0.1400
F_7	99%	3415	0.3195	100%	1208	0.1879
F_8	52%	1158	0.11186	60%	2084	0.3301
F_9	46%	2065	0.2026	50%	1251	0.2081
F_{10}	65%	3435	0.3321	81%	4012	0.6398

Table 1:Evaluation of success rate and computing time of 10-dimensional functions

Function	Nelder Mead's simplex method			Improved simplex method with quasi gradient		
	Success rate	Iteration	Computing time	Success rate	Iteration	Computing time
F_1	9%	2739	0.2539	100%	1387	0.2172
F_2	13%	4492	0.4213	100%	1216	0.1927
F_3	12%	24832	2.3841	52%	3047	0.5078
F_4	100%	13406	1.5418	100%	2108	0.4105
F_5	Failure			55%	25165	3.9881
F_6	2%	11494	1.1187	100%	2389	0.3942
F_7	Failure			100%	3538	0.5598
F_8	4%	7285	0.74172	60%	8805	1.4860
F_9	10%	19891	1.9298	53%	3016	0.5028
F_{10}	Failure			19%	11228	1.8882

Table 2: Evaluation of success rate and computing time of 15-dimensional functions

Function	Nelder Mead's simplex method			Improved simplex method with quasi gradient		
	Success rate	Iteration	Computing time	Success rate	Iteration	Computing time
F_1	Failure			100%	2735	0.4529
F_2	Failure			100%	1921	0.3207
F_3	Failure			44%	6618	1.1708
F_4	Failure			100%	4197	0.8969
F_5	Failure			54%	36073	6.0026
F_6	Failure			100%	4537	0.80412
F_7	Failure			100%	8204	1.3628
F_8	Failure			27%	17517	3.2283
F_9	Failure			40%	6559	1.1572
F_{10}	Failure			5%	12114	2.1718

Table 3: Evaluation of success rate and computing time of 20-dimensional functions

In experiments tested on PC with Intel Quad, the improved algorithm with quasi gradient method shows its better performance than Nelder Mead's simplex method in terms of success rate and computing time. When the scale of optimizing problems become larger, Nelder Mead's simplex method gets less success rate. With the same random choice of initial vertices, the improved simplex method always obtains higher convergence rate, less computing time than Nelder Mead's simplex method. It means that the improved simplex method here is more reliable and more effective in optimization than the original simplex method.

5. Applications

In the previous sections, the improved simplex method was presented. This algorithm has shown its better performance in several benchmark optimization functions. This section will apply this new algorithm in synthesizing lossy filters and training neural networks to control robot arm kinematics

5.1 Synthesis of lossy ladder filters

Ladder filters are made up of inductors and capacitors and widely used in communication systems. How to design a good filter with a desired frequency response is a challenging task because the traditional algorithms as Butterworth, Chebychev or inverse Chebychev, etc just synthesize filters without affects of lossy inductors and capacitors (Fig. 3). Therefore, frequency responses of these ideal filters are much different from the ones of real filters. In order to implement a real filter with lossy elements having similar characteristics as a prototype filter, we have to shift pole locations of the real filter close to pole locations of the prototype filter. In this part, the improved algorithm can be used to replace for analytical solutions which are very complex and inefficient especially with high order filters.

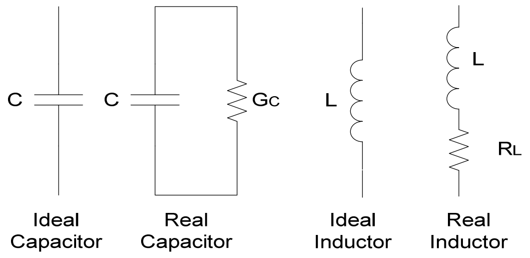

Fig.3. Models of real capacitor and real inductor

Our example is to design a 4th low-pass Chebychev filter with attenuation in pass-band α_p= 3dB, attenuation in stop-band α_s= 30 dB, pass-band frequency ω_p= 1Khz, stop-band frequency ω_s=2Khz. Using Chebychev methodology with ideal elements [22] we can find transfer function of this filter (eq. 6) and its circuit with all element values (Fig. 4).

$$H(S) = \frac{0.17198}{S^4 + 0.88598S^3 + 1.22091S^2 + 0.64803S + 0.17198} \tag{6}$$

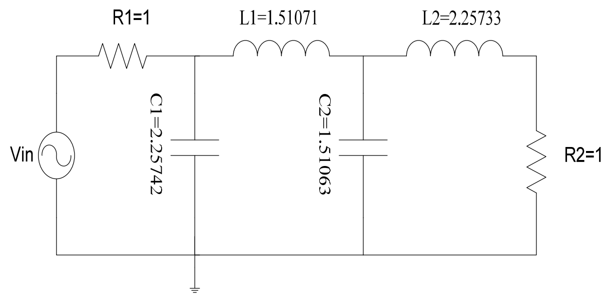

Fig.4. Circuit of ideal low-pass Chebychev filter

Instead of using ideal elements, we replace them by the lossy elements respectively and assume that CG1= CG2= RL1=RL2= 0.1 (Fig. 5). On account of this affect, the synthesized filter has the new transfer function (eq. 7) with different pole locations (Fig. 7). Therefore, frequency response of the real filter is changed with its shifted cut-off frequency to the left (Fig. 6).

$$H(S) = \frac{1}{a_1 S^4 + a_2 S^3 + a_3 S^2 + a_4 S + a_5} \tag{7}$$

where:

$$\begin{cases} a_1 = C_1C_2L_1L_2R_2 \\ a_2 = C_1L_1L_2 + 0.1C_1C_2L_1R_2 + 0.1C_1C_2L_2R_2 \\ + 0.1C_1L_1L_2R_2 + 0.1C_2L_1L_2R_2 + C_1C_2L_2R_1R_2 \\ a_3 = 0.1C_1L_1 + 0.1C_1L_2 + 0.1L_1L_2 + C_1L_2R_1 \\ + 0.01C_1C_2R_2 + 1.01C_1L_1R_2 + 1.01C_2L_1R_2 \\ + 0.01C_1L_2R_2 + 1.01C_2L_2R_2 + 0.01L_1L_2R_2 \\ + 0.1C_1C_2R_1R_2 + 0.1C_1L_2R_1R_2 + 0.1C_2L_2R_1R_2 \\ a_4 = 0.01C_1 + 1.01L_1 + 1.01L_2 + 0.1C_1R_1 + 0.1L_2R_1 \\ + 0.101C_1R_2 + 0.201C_2R_2 + 0.201L_1R_2 + 0.101L_2R_2 \\ + 1.01C_1R_1R_2 + 1.01C_2R_1R_2 + 0.01L_2R_1R_2 \\ a_5 = 0.201 + 1.01R_1 + 1.0301R_2 + 0.201R_1R_2 \end{cases} \tag{8}$$

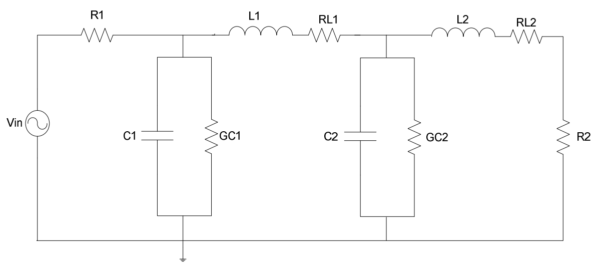

Fig.5. Circuit of lossy low-pass Chebychev filter

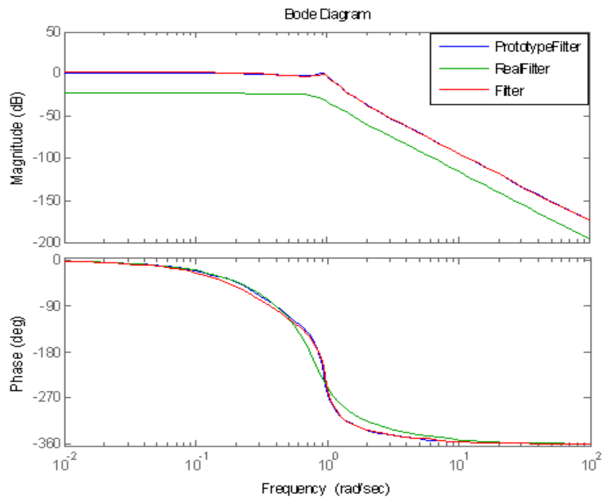

Fig.6. Frequency responses of filters (normalized)

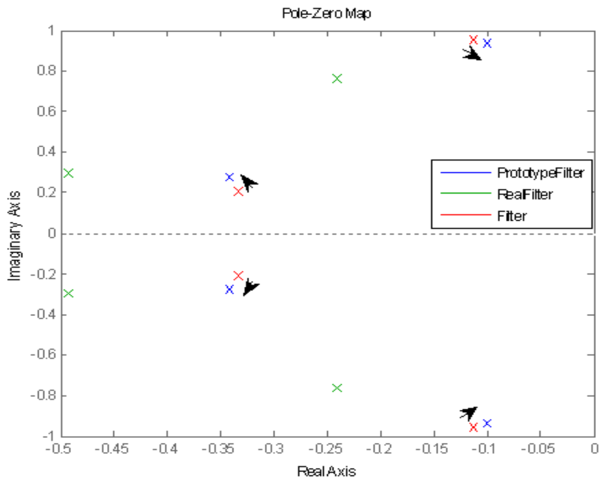

Fig.7. Pole locations of filters

In order to design a filter having desired characteristics as ideal filter, its transfer function (eq. 7) has to be similar to the transfer function of the prototype filter (eq. 6). In other words, its pole locations have to be close to pole locations of the ideal filter (Fig. 7). According to analytical method presented in [23], they have to solve a nonlinear system (eq. 9) of five equations with five unknowns (assume R_2 is known). Clearly, the analytical method is not an effective way to find proper solutions for this filter. Firstly, it is not easy to solve this nonlinear system especially with high order filters. Secondly, its solutions may not be applied in real implementation if one of component values is a negative number.

$$\begin{cases} a_1 = 1 \\ a_2 = 0.88598 \\ a_3 = 1.22091 \\ a_4 = 0.64803 \\ a_5 = 0.17198 \end{cases} \tag{9}$$

As presented in previous sections, the improved simplex method has ability to optimize high dimensional problems with very reliable convergence rate. For this particular application, this algorithm can be applied very effectively to synthesize filters. Instead of using the analytical method, we can use the improved simplex method to optimize the error function (eq.10) between coefficients of two transfer functions (eq. 6, 7). To guarantee reasonable results with all positive values, we may divide the numerator and denominator of the transfer function of the real filter by the value of C_1 in this case (which does not change characteristics of filters). The desired filter with R_1= 0.07131, R_2= 0.2, C_1= 3.3831, L_1= 0.70676, C_2= 9.7949, L_2= 0.7189 has similar frequency response and pole locations as the ideal filter (Fig. 6, 7).

$$Er=[(a_1-1)^2+(a_2-0.8859)^2+(a_3-1.2209)^2 \\ +(a_4-0.6480)^2+(a_5-0.1719)^2]/5 \tag{10}$$

5.2 Training neural networks to control robot arm kinematics

A neuron is normally connected by inputs, weights and a bias weight. Its output is defined by the standard activation function $f=(1+e^{-a})^{-1}, a=\sum_i^n (w_i x_i - w_{bias})$ while x_i is the input, w_i is the weight and w_{bias} is the bias weight. Each neuron can be connected together to form a neural network. A neural network is trained by input patterns and desired outputs. The weights of a network are adjusted to minimize the error between actual outputs and desired outputs. The error function is $E(w)=\frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left(\hat{y}_{m,p}-y_{m,p}\right)^2$, where $\hat{y}_{m,p}$ and y_m^p are the actual and desired outputs of a network respectively, P is the total number of patterns and M is the total number of output neurons.

Although neural networks showed their potential power for many applications but it is so difficult to train neural networks successfully. This frustration comes from the architecture of neural networks and the training algorithms. If the size is too small, neural networks cannot be trained. Inversely, if the size is too large, outputs from neural networks maybe not satiable. It means that neural networks can be used once we can select good architectures and good algorithms to train it. Actually, some algorithms are good at training these types of neural network architectures but are not good at training the others. In other words, it is not easy to find a reliable algorithm which has ability to train all types of neural network. Error Back Propagation (EBP) is considered as a breakthrough to train neural networks but it is not an effective algorithm because of its slow convergence. Lavenberg Marquardt is much faster but it is not suitable for large networks. Although these two algorithms are well-known, they usually face difficulties in many real applications because of their complex computations. Training neural networks to control robot arm kinematics is a typical example.

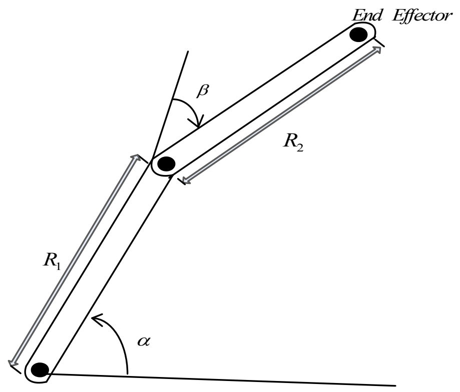

Fig. 8: Two-link planar manipulator

Forward kinematics is a practical example which can be resolved by neural networks. Neural networks can be trained to determine the position x and y of robot arms based on the data α, β read from sensors at the joints. This data set can be calculated from the following equations while R_1, R_2 are the fixed length arms and α, β are the movement angles of robot arms as shown in Fig. 8. By sensing its movement angles α, β, the position x and y of a robot arm can be determined.

$$x = R_1 \cos\alpha + R_2 \cos(\alpha+\beta) \tag{11}$$

$$y = R_1 \sin\alpha + R_2 \sin(\alpha+\beta) \tag{12}$$

To train a neural network with three neurons fully cascaded in Fig. 9, we use 2500 (50x50) training patterns generated from equations (11) and (12) with parameters α, β uniformly distributed in the range of [0, π] and R_1=R_2=0.5. The desired outputs and the actual outputs from this network are depicted in Fig. 10, 11.

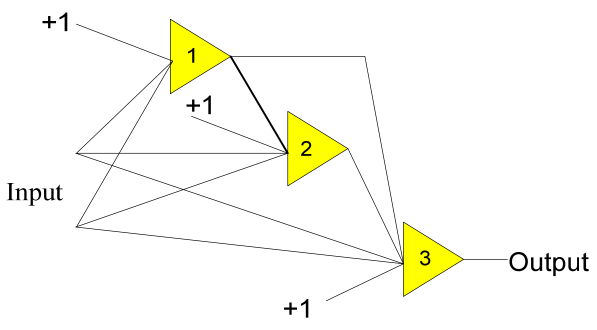

Fig. 9: Neural network architecture to control robot arm kinematics

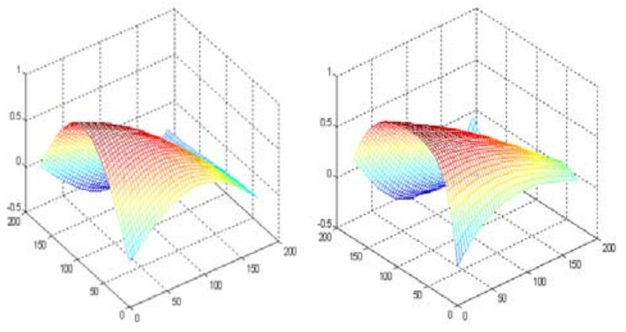

Fig. 10: Desired output and actual output from neural network in x direction

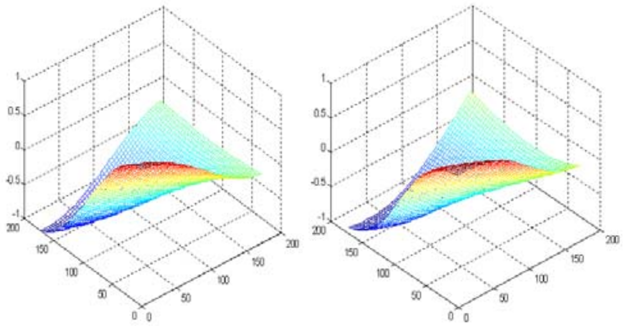

Fig. 11: Desired output and actual output from neural network in y direction

As we can see, the desired outputs and actual outputs from the neural network are not so much different with an error about 0.001. The advantage of this algorithm in training neural networks is that its computational cost is proportional to the number of weights not the number of input patterns. For this particular case, all 2500 patterns can be applied at once, and the improved simplex method will train neural networks by optimizing a function with seven variables which are equal to seven weights. In contrast, training neural networks with Error Back Propagation for 2500 patterns seems impractical because EBP has to adjust its weights for each training pattern in each iteration. This training process is very time-consuming and inapplicable for this particular case. Levenberg Marquardt is known as a fast training algorithm but its training ability is limited by the number of input patterns P, weights N, and outputs M. In other words, the problem becomes more difficult with increase of the size of the network. In each iteration, this algorithm has to calculate the Jacobian matrix $J_{P*M \times N}$ and the inversion of J^TJ square matrix. It is obvious that LM cannot train neural networks with seven weights and 2500 input patterns for this robot arm kinematics because of the huge size of Jacobian matrix $J_{2500\times7}$ which over-limits computing capability of PC computers. In order to train neural networks with EBP or LM, the size of training patterns usually has to be reduced. This will ease the computing tension but it will affect the accuracy of neural network outputs significantly. Therefore, the actual outputs may be much different from the desired outputs. In contrast, the increased size of input patterns may affect the convergence rate but not the training ability of the improved simplex method. This character makes it different from Error Back Propagation and Lavenberg Marquardt. The improved simplex method can be a useful algorithm to train neural networks for many real applications.

6. Conclusions

The improved simplex algorithm is presented in this paper. This algorithm is tested over several benchmark optimization problems and shows its better performance compared with Nelder Mead's simplex method in terms of success rate or computing time. This algorithm shows a great deal of large scale optimization. This algorithm also shows its ability in many real applications which other algorithms or other analytical methods maybe not suitable ore ineffective because of their computing cost or complexity. Synthesizing lossy ladder filter or training neural networks to control robot arm kinematics with the improved simplex method are two typical examples presented in this paper. More methods of calculating quasi gradients with higher accuracy and more experiments which will be tested are our future researches.

References

[1] J. A. Nelder, R. Mead, "A Simplex Method for Function Minimization", *Computer Journal*, vol. 7, pp. 308-313, 1965.

[2] I. Jordanov, A. Georgieva, "Neural Network Learning with Global Heuristic Search", *IEEE Trans. on Neural Networks*, vol. 18, no. 3, pp. 937-942, 2007.

[3] S. Shrivastava, K. R. Pardasani, M. M. Malik, "SVM Model for Identification of human GPCRs", *Journal of Computing*, vol. 2, no. 2, pp. , 2010.

[4] G. Guo, Y. Shouyi, "Evolutionary Parallel Local Search for Function Optimization", *IEEE Trans. on System, Man, and Cybernetics*, vol. 33, no. 6, pp. 864-876, 2003.

[5] D. Stiawan, A. H. Abdullah and M. Y. Idris, "Classification of Habitual Activities in Behavior-based Network Detection", *Journal of Computing*, vol. 2, no. 8, pp. , 2010.

[6] E. Öz, İ. Guney, "A Geometric Programming Approach for The Automotive Production in Turkey", *Journal of Computing*, vol. 2, no. 7, pp. , 2010.

[7] H. Miyamoto, K. Kawato, T. Setoyama, and R. Suzuki, "Feedback-Error Learning Neural Network for Trajectory Control of a Robotic Manipulator", *IEEE Trans. on Neural Networks*, vol. 1, no. 3, pp. 251–265, 1988.

[8] Y. Fukuyama, Y. Ueki, "An Application of Neural Networks to Dynamic Dispatch Using Multi Processors", *IEEE Trans. on Power Systems*, vol. 9, no. 4, pp. 1759-1765, 1994.

[9] S. Sadek, A. Al-Hamadi, B. Michaelis, U. Sayed, "Efficient Region-Based Image Querying ", *Journal of Computing*, vol. 2, no. 6, pp. , 2010.

[10] D. E. Rumelhart, G. E Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors", *Nature*, vol. 323, pp. 533-536, Oct. 9, 1986.

[11] M. T. Hagan, M. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm", *IEEE Trans. on Neural Networks*, vol. 5, no. 6, pp. 989-993, 1994.

[12] K. Levenberg, "A Method for the Solution of Certain Problems in Least Squares," *Quart. Appl. Mach*., vol. 2, pp. 164–168, 1944.

[13] B. M. Wilamowski, Hao Yu, "Improved Computation for Levenberg-Marquardt Training", *IEEE Trans. on Neural Networks*, vol. 21, no.6, pp. 930-937, June. 2010.

[14] R. A. Jacobs, "Increased Rates of Convergence through Learning Rate Adaption", *IEEE Trans. on Neural Network*, vol. 5, no. 1, pp. 295-307, 1988.

[15] T. Tollenaere, "SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties", *IEEE Trans. on Neural Networks*, vol. 3, no. , pp. 561-573, 1990.

[16] R. Salomon, J. L. Van Hemmen, "Accelerating Back Propagation through Dynamic Self-Adaption", *IEEE Trans. on Neural Networks*, vol. 9, no. , pp.589-601, 1996.

[17] M. Manic, B. M. Wilamowski, "Random Weights Search in Compressed Neural Networks Using Over-determined Pseudoinverse", *IEEE International Symposium on Industrial Electronics 2003*, vol. 2, pp. 678-683, 2003.

[18] L. Nazareth, P. Tseng, "Gilding the Lily: A Variant of the Nelder-Mead Algorithm Based on Golden-Section Search", *Comput. Optim. Appl*, vol. 22, no. 1, pp. 133–144, 2002.

[19] F. Gao, L. Han, "Implementing the Nelder-Mead Simplex Algorithm with Adaptive Parameters", *Comput. Optim. Appl*, vol., no. , pp. , 4th May 2010.

[20] Torczon, V.:Multi-directional Search: A Direct Search Algorithm for ParallelMachines. Ph.D. Thesis, Rice University, TX (1989).

[21] J. T. Betts, "Solving the Nonlinear Least Square Problems: Application of a General Method", *Journal of Optimization Theory and Applications*, vol. 8, no. 4, 1976.

[22] B. M. Wilamowski and R. Gottiparthy "Active and Passive Filter Design with MATLAB", *International Journal on Engineering Educations*, vol. 21, No 4, pp. 561-571, .2005

[23] Marcin Jagiela and B.M. Wilamowski "A Methodology of Synthesis of Lossy Ladder Filters", 13th *IEEE Intelligent Engineering Systems Conference*, INES 2009, Barbados, April 16-18., 2009, pp. 45-50, 2009.

Nam D. Pham (S'08) received the M.S. degree in electrical engineering from Auburn University, Auburn, AL, where he is currently working toward the Ph. D degree in electrical engineering.He is a Research Assistant with the Department of Electrical and Computer Engineering, Auburn University. His main interests include numerical optimization, neural networks, database systems and network security.

Bogdan M. Wilamowski (M'82– SM'83–F'00) received the M. S. degree in computer engineering, the Ph.D. degree in neural computing, and the Dr. Habil. degree in integrated circuit design in 1966, 1970, and 1977, respectively. He received the title of Full Professor from the President of Poland in 1987. He was the Director of the Institute of Electronics (1979–1981) and the Chair of the Solid State Electronics Department (1987–1989), Technical University of Gdansk, Gdansk, Poland. He was/has been a Professor with the Gdansk University of Technology, Gdansk (1987–1989), the University of Wyoming, Laramie (1989–2000), the University of Idaho, Moscow (2000–2003), and Auburn University, Auburn, AL (2003–present), where he is currently the Director of the Alabama Micro/Nano Science and

Technology Center and a Professor with the Department of Electrical and Computer Engineering. He was also with the Research Institute of Electronic Communication, Tohoku University, Sendai, Japan (1968–1970), and the Semiconductor Research Institute, Sendai (1975–1976), Auburn University (1981–1982 and 1995–1996), and the University of Arizona, Tucson (1982–1984). He is the author of four textbooks and about 300 refereed publications and is the holder of 28 patents.