論文 / 著書情報
Article / Book Information

論題(和文)	
Title(English)	IIR Adaptive System Identification based on Particle Swarm Optimization with Improved Cost Function
著者(和文)	YUENYONG SUMETH, 西原明法
Authors(English)	Sumeth Yuenyong, AKINORI NISHIHARA
出典(和文)	第26回信号処理シンポジウム, , , 676-679
Citation(English)	26th SIP Symposium, , , 676-679
発行日 / Pub. date	2011, 11
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2011 Institute of Electronics, Information and Communication Engineers.

IIR Adaptive System Identification based on Particle Swarm Optimization with Improved Cost Function

Sumeth Yuenyong and Akinori Nishihara

Department of Communication and Integrated Systems
Tokyo Institute of Technology
Email: sumeth@nh.cradle.titech.ac.jp

***Abstract*—This work explores the use of Particle Swarm Optimization (PSO) as an alternative to training adaptive IIR filters in system identification setting with focus on the estimation of the cost function to be minimized. It was found that the usual way to estimate the cost function when PSO is applied to adaptive filtering has a problem, the cost function values are noisy, due to inaccurate early samples of block-error signal used to form the mean square error. This causes a problem when particles move close together, as the cost function value can change slightly, even for the same location in parameter space. This prevents the algorithm from properly conducting fine search of the filter coefficient space, as all points in a small region have ambiguous cost function values due to noise. A modification was proposed to address this problem by extending the window size and discarding early samples of the error signal. Experiments using the standard constriction-factor PSO was conducted to test the proposed modification against the standard. Due to different cost function estimation methods, an independent mean square error value, calculated separately from the cost functions, was used as performance evaluation. The results demonstrate improved performance in terms of lower mean squared error values by 5 dB for high order filter with noise at the plant's output, and several orders of magnitude for second order filter with no noise.**

I. Introduction

System identification tasks that deal with systems which are recursive in nature require the use of adaptive IIR filters in order to model the transfer function of the unknown plant more accurately than possible with FIR filters. However, IIR adaptive filters are more difficult to train due to the fact that the error performance surface is multi-modal under certain conditions. This means that gradient based training algorithms can become stuck in a local minimum, moreover, they are generally slower to converge compared with their FIR counterparts [1]. These drawbacks motivate the use of global optimization algorithms to train adaptive IIR filters.

Particle Swarm Optimization (PSO) algorithm is a global optimization algorithm proposed by Kennedy in [2], after which there have been proposed many modifications to the algorithm [3]. In this work we used the standard constriction factor PSO [4] as a test platform. The concepts of PSO is that the algorithm starts by generating M random points in the weight (coefficient) space of the filter, each point is called a particle, and the set of all particles is called the swarm. In each iteration, the value of the cost function (the unknown error performance surface of the filter) [5] at each point is estimated. Each particle are then given velocity according to the distances to two points in space: the point where the lowest cost function value had been encountered by that particle, called personal best or pbest, and another point where the lowest cost function value had been encountered by any particle, called global best or gbest. Particles will tend to move in general directions toward these two points. If a lower cost function value than pbest or gbest is found by a particle, then these points get updated. Heuristically, concentrating the particles around the region in space where the lowest cost function value had been found increases the chance of finding a point with lower cost value. This way, the weight space is efficiently searched and a point that enables satisfactory performance of the filter likely can be found.

Denoting the iteration number as n, pbests and gbest are updated by

$$\hat{\mathbf{x}}_i(n) = \begin{cases} \hat{\mathbf{x}}_i(n-1) & J(\mathbf{x}_i(n)) \geq J(\hat{\mathbf{x}}_i(n-1)) \\ \mathbf{x}_i(n) & J(\mathbf{x}_i(n)) < J(\hat{\mathbf{x}}_i(n-1)) \end{cases}, \tag{1}$$

$$J(\hat{\mathbf{x}}_i(n)) = \begin{cases} J(\hat{\mathbf{x}}_i(n-1)) & J(\mathbf{x}_i(n)) \geq J(\hat{\mathbf{x}}_i(n-1)) \\ J(\mathbf{x}_i(n)) & J(\mathbf{x}_i(n)) < J(\hat{\mathbf{x}}_i(n-1)) \end{cases}, \tag{2}$$

$$J(\mathbf{g}(n)) = \min_i(J(\hat{\mathbf{x}}_i(n)), \tag{3}$$

$$\mathbf{g}(n) = \operatorname*{argmin}_i(J(\hat{\mathbf{x}}_i(n)), \tag{4}$$

where $\mathbf{x}_i, \hat{\mathbf{x}}_i, \mathbf{g}$ and $J()$ denote respectively the current position and pbest of particle i, gbest, and the cost function. The next step is to update the velocities and positions of particles by

$$\mathbf{v}_i(n+1) = K[\mathbf{v}_i(n) + c_1\mathbf{D}(\hat{\mathbf{x}}_i - \mathbf{x}_i) + \mathbf{D}(\mathbf{g} - \mathbf{x}_i)], \tag{5}$$

$$\mathbf{x}_i(n+1) = \mathbf{x}_i(n) + \mathbf{v}_i(n+1), \tag{6}$$

where,

$$\varphi = c_1 + c_2, \tag{7}$$

$$K = \begin{cases} \frac{2}{-2+\varphi\sqrt{\varphi^2-4\varphi}} & \varphi > 4 \\ K_0 & \text{otherwise} \end{cases}. \tag{8}$$

The variables c_1 and c_2 are the acceleration coefficients which are parameters of the PSO algorithm and $\mathbf{D}$ is an $L \times L$

diagonal matrix whose entries are uniform random numbers in the range 0 to 1 and L is the number of filter coefficients. The constant K is the "constriction factor", which keeps the velocities from increasing without bound and causing the algorithm to become unstable.

II. Estimating the Cost Function

Unlike gradient based methods, which need only the current error sample $e(n)$ of the filter in order to update the filter weights, global methods such as PSO need accurate value of the cost function at each point, i.e., $E[e^2(n)]$ needs to be estimated instead of just using $e(n)$ as the cost function to be optimized. This is because all global methods progress from rough search to fine search as the number of iterations increases, using only $e(n)$ which is an inaccurate representation of $E[e^2(n)]$ causes error in the rough search stage and subsequently fine search will be conducted in the wrong region, leading to poor result. The ensemble average, $E[e^2(n)]$ has to be estimated by error samples, using the notation that $J(\mathbf{x}_i)$ denotes the cost function value at position $\mathbf{x}_i$ in the weight space from (2), we have

$$J(\mathbf{x}_i(n)) = E_i[e^2(n)] \approx \frac{1}{Q}\sum_{k=0}^{Q-1} e_i^2(n-k), \tag{9}$$

where i is the index for the i$^{\text{th}}$ particle, $e_i(n-k)$ is the error produced by a filter whose weights are represented by that particle, and Q is the number of samples involved in the estimation, which gets more accurate as Q becomes larger. In practice, Q is limited by computational complexity, in fact, most of the computational load of training adaptive filter with a global method comes from estimation of the cost function.

The expression in (9) is deceptively simple. It looks like a moving average, which incorrectly implies that to estimate $E_i[e^2(n)]$ one simply need to keep the previous $N-1$ samples of $e_i(n)$ produced by each particle i in each iteration and then take the squared average. However, this is the wrong way to estimate the cost function, because the samples $e_i(n-1), e_i(n-2), ..., e_i(n-Q+1)$ were all calculated using different filter weights (because each belong to different iterations of the algorithm, and the weights are updated every iteration) and hence these error values are not representative of the current position in the weight space whose cost function value is being estimated. In fact, only one sample, $e(n)$ was calculated using the weights at the current position of the particle.

The correct way of estimating the cost function is to keep the input samples $x(n), x(n-1), ..., x(n-Q+1)$ in memory and *recompute* the output samples $y(n), y(n-1), ..., y(n-Q+1)$ for *every iteration* of the algorithm. This way the error samples $e(n), e(n-1), ..., e(n-Q+1)$ in (9) are all calculated from the same filter weights and therefore they form accurate estimate of how good the weights are in predicting the unknown system. This way of estimating the cost function is shown by the block diagram in Figure 1. In this figure, it can be seen that the input samples are stored within a buffer of length

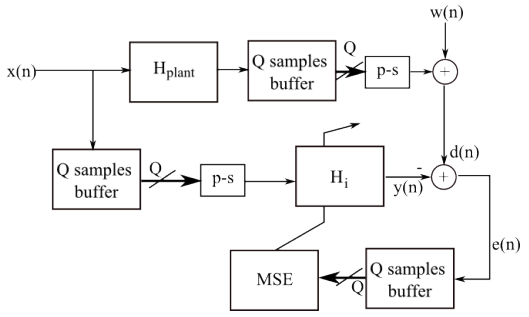

Fig. 1. Estimating The Cost Function

Q on the left-hand side. For each iteration of the algorithm, the outputs due to particle i is calculated using the stored input samples by filtering with the filter represented by H_i (particle i), which are compared with the corresponding time samples output from the plant to calculate the error samples that form the estimate of $J(\mathbf{x}_i)$. This process is repeated for each particle in order to update pbests and gbest that the PSO algorithm is based on. Additionally, each of the H_i must have its internal signals cleared every iterations, since particles can move very far (compared to gradient update) and using the signals generated by the previous iteration will affect the output of the current iteration.

A. Issue with The Cost Function

The cost function value estimated as in Figure 1 has an issue. That is, the output of the plant is obtained by continuous filtering without clearing of internal signals, while each of H_i always start filtering with cleared internal signal. This means that the early samples of the error will have large magnitudes. Thus, when taking the average mean squared error, these early samples will dominate and "obscure" the later samples, which are in fact more important since the internal signals of H_i had been completely loaded and these later samples more accurately reflect value of $J(\mathbf{x}_i)$

B. Proposed Modification

In order to deal with the aforementioned issue, this work proposes a simple modification to the way the cost function is calculated. Instead of using all Q error samples to form the cost function estimate according to (9), one simply discard some early samples of $e(n)$ and use only the remaining samples to form the estimate. That is

$$J(\mathbf{x}_i(n)) \approx \frac{1}{Q_1}\sum_{k=0}^{Q_1-1} e_i^2(n-k), \tag{10}$$

where $Q_1 < Q$. It should be noted here that all of the Q samples had to be calculated. Due to the recursive nature of IIR filters, the difference between Q and Q_1 had to be quite large compared to the order of the filter, because the effect of the early output propagate forward due to feedback. From experiments, it was found that using $Q = 130$ and $Q_1 = 100$ works well, irrespective of other parameters.

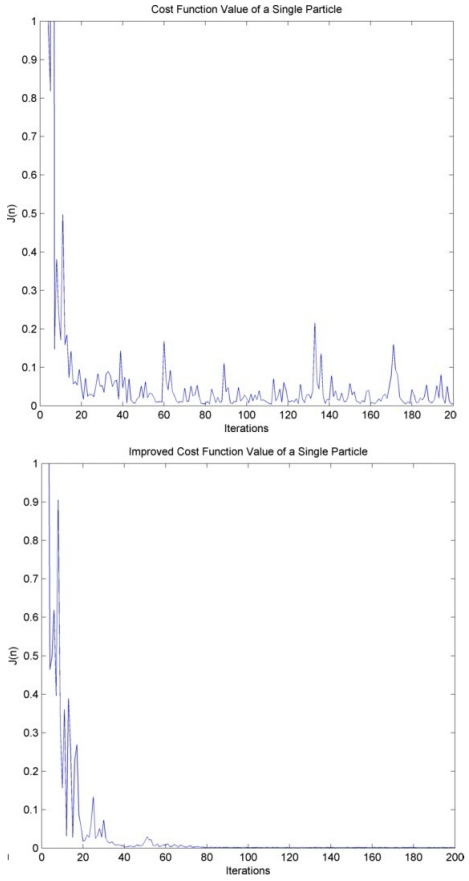

Fig. 2. Old and Improved Cost Function Values Over 200 Iterations

In order to illustrate the advantage of (10) over (9), two plots comparing the cost function values of a single particle over 200 iterations is shown in Figure 2. The top panel shows the cost function estimated using (9), it can be seen that the value of the cost function still varies significantly during the later iteration, which by nature of the PSO algorithm should not happen because velocities of particles should now be very small. This fluctuation in the cost function values is the noise that prevents the algorithm from conducting fine search during the later iterations.

The bottom panel of the figure showed the cost function values estimated using (10). It can be seen that the cost function values no longer vary significantly during the later iterations. This allows the algorithm to conduct fine search, and lead to relatively better result.

III. Experiment Results

The experiments followed standard system identification configuration with white noise input. The performance of two schemes based on different cost function estimates has to be compared by defining a common cost function that is used to evaluate both schemes. In each iteration, the quality of the gbest position is evaluated by using its weights to filter a white noise sequence (independently generated and different from the input $x(n)$) of length 1000 samples, and comparing the output with the output of the plant fed by the same input using the mean squared error criterion. The parameters of the algorithm fixed for all experiments: $c_1 = 2.2$, $c_2 = 1.9$, $M = 20$, $Q = 130$ and $Q_1 = 100$. Each experiment was run for 100 trials, for each trial, the MSE value of each iteration was calculated as described previously to form a curve. The average of the 100 MSE curves were then taken across the trials to yield to final MSE curve. The output of the plant had additive white noise with power of -30 dB to simulate measurement noise. The adaptive filter had lattice form, so that it was easy to initialized particles within the stable region by simply restricting the reflection coefficients to be in the range [-1 1].

A. Matched Order Filter with Noise Floor

The unknown plant is given by the ladder-lattice coefficients [6]

$$k_1,k_2,k_3,k_4,k_5 = 0.4621, 0.0660, 0.2798, 0.01001, 0.0113 \quad (11)$$

$$v_0,v_1,v_2,v_3,v_4,v_5 = -0.0647, -0.1910, 0.0941, 0.5499, 0.4351, 0.1057 \quad (12)$$

with a matched order filter with exactly the same number of coefficients. The result is shown in Figure 3, the dashed curve indicate the performance of gradient descend for comparison, blue curve is the PSO with unmodified cost function, and red curve is the PSO with the proposed cost function. The final MSE value of the red curve was 5 dB lower than the blue one, and both were lower than gradient descend over 200 iterations. The reason why they are not able to reach the noise floor was due to the limitation of the algorithm itself. As with any stochastic search algorithm, PSO performs best in low dimensional space, as the volume of space to be searched is less. Since this volume increases exponentially with the number of dimensions, higher order filter such as (12) is a challenge for the algorithm.

B. Matched 2nd Order Filter

The unknown plant is given by [7]

$$k_1, k_2 = -0.0966, 0.5132 \quad (13)$$

$$v_0, v_1, v_2 = 0.1240, -0.3744, 0.1750 \quad (14)$$

In this experiment, the adaptive filter was second order lattice form with coefficients $\{k_1, k_2, v_0, v_1, v_2\}$. The noise at the output of the plant was removed, and the result is shown in Figure 4. In this case, the difference between the two cost functions becomes several orders of magnitude. The PSO with improved cost function was able to identify the system almost perfectly. From the experiment results, it can be concluded that the proposed modification allows the algorithm to perform to its full potential in situations where it can perform well, which are for low order plants. As can be seen from the result in Figure 4, without the modification, the performance in the ideal case with low order plant and no noise was not much better than that of high order with noise. This demonstrates the importance of the modification.

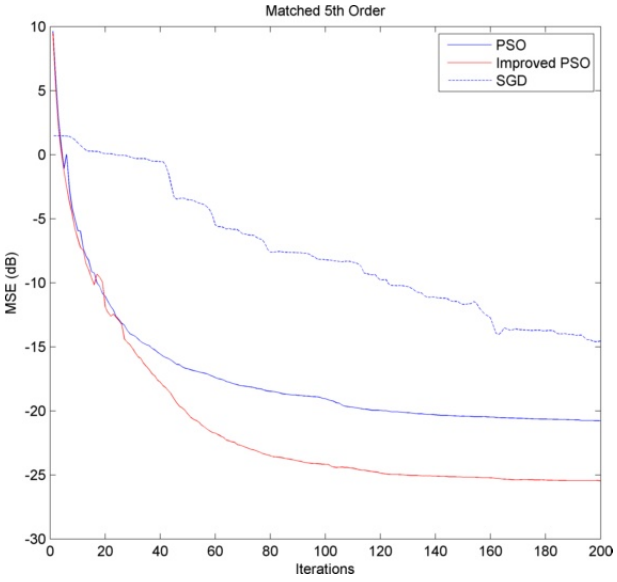

Fig. 3. MSE Curve for Identification of (12)

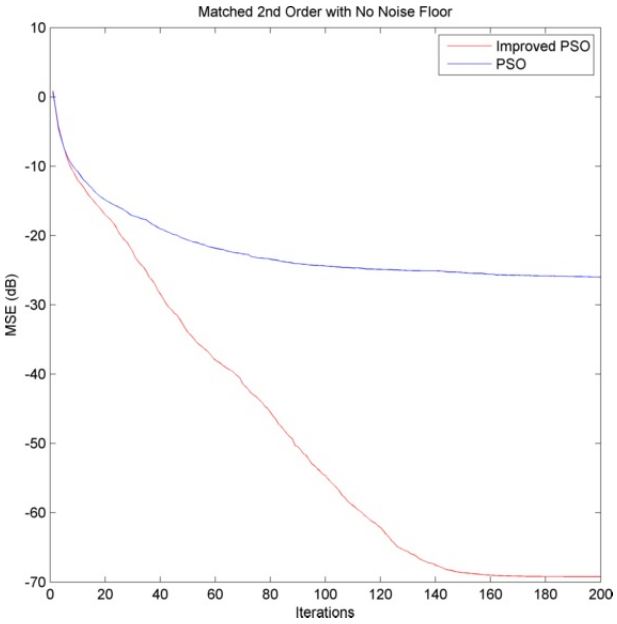

Fig. 4. Matched Second Order with No Noise Floor

IV. Conclusion

A new way to estimate the cost function when PSO algorithm is applied to training adaptive filter for system identification application was proposed. The improved cost function is less noisy in a sense that its value does not fluctuate much when particles move slower during later iterations. This allows the algorithm to conduct fine search better and resulted in better performance as demonstrated by the experiments. In lower order system with the noise floor removed, using (10) allowed the algorithm to identify the plant very accurately. Therefore, the proposed cost function is promising, and further research will be addressed at overcoming the limitation of algorithm with higher order systems by using cooperative learning approach [8].

References

[1] P. Regalia, *Adaptive IIR Filtering in Signal Processing and Control*, 1st ed. CRC Press, Oct. 1994.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, 1995, pp. 1942–1948 vol.4.

[3] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.

[4] Y. Shi and R. Eberhart, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, 2000, pp. 84–88.

[5] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice Hall, 2002.

[6] D. Krusienski and W. Jenkins, "Adaptive filtering via particle swarm optimization," in *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, vol. 1, 2003, pp. 571–575 Vol.1.

[7] N. Norman S., *Control System Engineering*, 6th ed. John Wiley & Sons, 2011.

[8] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 225 – 239, Jun. 2004.