PARAMETER ESTIMATION FOR TIME-VARYING SYSTEM BASED ON COMBINATORIAL PSO

Weixing Lin[1, 2], Peter X. Liu[2]

[1)] *Faculty of Information Science and Technology, Ningbo University, Ningbo, Zhejiang Province, China 315211*

[2)] *Department of System and Computer Engineering, Carleton University, Ottawa, Ontario, Canada K1S 5B6*

wlin@sce.carleton.ca

In this paper, a novel Particle Swarm Optimization (PSO) identification algorithm for time-varying systems with a colored noise is presented. Presented criterion function can show not only outside system output error but also inside parameters error in order to explain more difference between actual and estimative system. Identification algorithm may consist of many different PSO algorithms that are named the combinatorial PSO. The estimating and tracking of parameters make use of characteristics of different PSO algorithms. The simulation and result show that the identification algorithm for time-varying systems with noise was indeed more efficient and robust in combinatorial PSO comparing with the original particle swarm optimization.

1. INTRODUCTION

For time-varying system identification and parameter tracking is still a very active research field in the recent years [1]. Most of identification algorithms are still the Recursive Least Squares (RLS) or gradient algorithms, which search along the direction of parameters change slowly. If the search space is not differentiable or the parameters are nonlinear, usually the optimal global solutions would not be found. In the algorithms the selection of the gain or forgetting factor becomes very important. In general, a large gain or a small forgetting factor makes the RLS or gradient algorithms to have a better ability for tracking the variation of parameters, but also makes them sensitive to white noise. On the other hand, a small gain or a big forgetting factor makes the algorithms less sensitive to white noise, but at the same time results in a poor tracking ability for slowly time-varying systems. There are many new identification algorithms such as identification using Genetic Algorithms (GA) in order to have good tracking ability and be not sensitive to white noise [2] [3]. GA can define search direction and scopes only based on the fitness function converted from the object function and doesn't need to know the differential of object function and other auxiliary information. This will be very convenient for the functions whose differentials are difficult to find or are not existent. Above algorithms use all models with white noise to study.

The particle swarm optimization (PSO) is similar to GA where the system is initialized with a population of random solutions. It is unlike GA, however, where every potential solution is also assigned a randomized velocity and the potential solutions called particles are "flown" through the search space. In this paper a novel PSO identification algorithm for time-varying systems with a colored noise is presented. Because PSO algorithm is a stochastic optimization algorithm as same as GA, it should be above advantages of GA.

2. OVER VIEW OF PARTICLES SWARM OPTIMIZATION

2.1 Features of Particle Swarm Algorithm

In 1995, Kennedy and Eberhart first introduced PSO method [4]. The PSO is a stochastic optimization technique that can be likened to the behavior of a flock of birds and that is derived from the social-psychological theory. The method has been found to be robust in solving problems featuring nonlinearity and no differentiability, multiple optimization, and high dimensionality through adaptation.

The PSO is a swarm based optimization technique. A simple explanation of the PSO's operation is as follows. Every particle represents a possible solution to the optimization task at hand. During iterations every particle accelerates in the direction of its own personal best solution found so far, as well as in the direction of the global best position discovered by the particles in swarm so far. This means that if a particle discovers a promising new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process.

In PSO every particle has the following attributes and is treated as a volume-less point in the D -dimensional search space. The *ith* particle can be described by a vector $X_i = [x_{i1}(t), x_{i2}(t) \Lambda\ x_{iD}(\mathrm{t})]^T$. The personal best position of the *ith* particle in the search space is recorded and represented as $P_i = [p_{i1}(t), p_{i2}(t)...p_{iD}(t)]^T$. The global best position found by any particle during previous steps is represented by the symbol P_g, i.e. $P_g = [p_{g1}(t), p_{g2}(t)...p_{gD}(t)]^T$. The flying velocity of particle i is represented as $V_i = [v_{i1}(t), v_{i2}(t)...v_{iD}(t)]^T$. The maximum velocity of particles is $V_{\max} = (v_{\max 1}, v_{\max 2}...v_{\max D})^T$, and the range of the particles is $\mathrm{X}_{\max} = (x_{\max 1}, x_{\max 2}...x_{\max D})^T$. Assuming that the function J_h is to be minimized, the velocity and position of every particle can be modified respectively by the following equations

$$v_{id}{}'(t+1) = w \times v_{id}(t) + C_1 \times rand_1 \times [p_{id}(t) - x_{id}(\mathrm{t})] + C_2 \times rand_2 \times [p_{gd}(t) - x_{id}(\mathrm{t})] \tag{1}$$

Where $i = 1,2 \Lambda\ N_1$, $d = 1,2 \Lambda\ D$. N_1 is the number of particles. C_1 and C_2 are acceleration coefficients that are positive constants. Acceleration coefficients control how far a particle will move in a single iteration. Typically, these are both set to 2.0, although assigning different values to and sometimes leads to improved performance [5]. $rand_1 \sim U(0,1)$ and $rand_2 \sim U(0,1)$ are two uniform random sequences in the range (0,1). The component value in every vector V_i can be clamped to the range $[-v_{\max d}, v_{\max d}]$ to reduce the likelihood of particles leaving the search space. The value of $v_{max\, d}$ is usually chosen to be $k \times X_{\max}$, with $0.1 \le k \le 1.0$ [6]. The formula of clamped velocity may show as

$$v_{id}(t+1) = \begin{cases} v_{\max d} & v_{id}'(t+1) > v_{\max d} \\ v_{id}'(t+1) & -v_{\max d} \le v_{id}'(t+1) \le v_{\max d} \\ -v_{\max d} & v_{id}'(t+1) < -v_{\max d} \end{cases} \tag{2}$$

The new position of a particle is calculated using

$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$ (3)

The personal best position of each particle is updated using

$$P_i(t+1) = \begin{cases} P_i(t), & J_h(X_i(t+1) \ge J_h(P_i(t)) \\ X_i(t+1), & J_h(X_i(t+1) < J_h(P_i(t)) \end{cases} \tag{4}$$

and the global best position found by any particle during previous steps, is defined as

$$P_g(t+1) = \arg \min_{P_i} J_h(P_i(t+1)) \quad 1 \le i \le N_1 \tag{5}$$

The first item of (1) is the momentum term and w is its inertia weight.

The inertia weight w is employed to control the impact of previous velocity on the current velocity in order to influence global and local exploration abilities of the “flying points”. A larger inertia weight facilitates global exploration while a smaller inertia weight tends to facilitate local exploration to the current search area. Suitable selection of the inertia weight can provide a balance between global and local exploration abilities and require less iteration on average to find the optimum. There is a kind of simple way the value w is linearly decreased with the iteration going. By means of the mathematics, the description is [8]

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} iter \tag{6}$$

Where $w_{\max}$ is the initial weight, $w_{\min}$ is terminative weight, $iter_{\max}$ is the biggest iteration times, $iter$ is the current iteration times.

Usually, we define the inertia weight w to linearly decrease from 0.9 to 0.4 during the iterations. Every particle is updated according to its own flying experience and the group’s flying experience. From simulation we know the particle maybe loss its optimum when the inertia weight w too large, thus the algorithm will be failed to converge. So far we do not know the best and accurate changing regulation of the inertia weight. To overcome this shortage, we induce fuzzy logic rules algorithm into PSO. The algorithm is the PSO with fuzzy self-adapting inertia weights (FSPSO).

Considering fuzzy logic rules and the last results of particle's searching, it modifies dynamically the values of inertia weight in order to promote the convergent capability of the particle swarm [7].

If we establish the variable J_h of the square sum of output residual reflected directly, the opposite quantity $\Delta\delta$ related directly with the square sum of the output residual is described. The mathematics formula may show as $\Delta\delta = \frac{J_h - J_{h\min}}{J_{h\max} - J_{h\min}}$, where J_{hmax} and J_{hmin} is the top and bottom limit that J_h may take the value respectively. In the formula we get the value $\Delta\delta$ placed in [0, 1] interval, which is more fit than J_h to be the input of the fuzzy reasoning machine. Two inputs of the fuzzy reasoning are defined as three fuzzy sets respectively. It is named as small, medium and big. Nine kinds of different combinations of inputs correspond with nine outputs to decide the next iteration's value w. We set the fuzzy self-adapting rule and mainly consider two points. One is that when J_h is big, the increment of the next iteration w is big so that particles can be in the big range searching, vice versa. The other one is that when the value of the current iteration w is big, the minus quantity of the next iteration w is big so that particles can accelerate the convergence. The rule table of fuzzy self-adapting algorithm is showed in Table 1. The simulation result shows that the strategy adjusting w can obtain the more satisfied result.

Recently, work by Clerc [9]-[11] indicated that a constriction factor may help to ensure convergence. Application of the constriction factor results in (7).

$$\begin{aligned} v_{id}'(t+1) = K[v_{id}(t) + C_1' \times rand_1 \times [p_{id}(t) \\ - x_{id}(\text{t})] + C_2' \times rand_2 \times [p_{gd}(t) - x_{id}(\text{t})]] \end{aligned} \tag{7}$$

where $K = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$ and $\varphi = C_1' + C_2'$, $\varphi \geq 4$.

If you compare (7) with (1) you will find that $w = K, C_1 = KC_1'$ and $C_2 = KC_2'$. For example: if $C_1' = C_2' = 2$, (7) is the same as (1) in w=1. It is seen that the values of w, C_1 and C_2 are important in PSO.

2.2 Combinatorial PSO

Eberhart and Shi [12] have shown that the constriction factor alone does not necessarily result in the best performance. Combining more approaches could result in the fastest convergence overall. These improvements appear to be effective on a large collection of problems.

Kennedy has taken this LBEST version of the particle swarm and applied to it a technique referred to as "social stereotyping" [13] [14]. A clustering algorithm is used to group individual particles into "stereotypical groups". The cluster center $G_i(t)$ is computed for every group and then substituted into (1), yielding three strategies to

Table 1 - The Rule table of The Fuzzy Self-adapting Algorithm

Inertial weight of the next iteration [0.4,0.9]		The opposite quantity directly related with the square sum of the output residual $\Delta\delta$, [0, 1]		
		Small [0,0.35]	Medium [0.35,0.7]	Big [0.7,1]
Inertial weight of the current iteration w	Small, [0.4,0.6)	0.4	w +0.08;	w +0.15
	Medium, [0.6,0.75)	w -0.05	w	w +0.10
	Big, [0.75.0.9]	w -0.10	w -0.08	w +0.05

calculate the new velocity

$$V_i^{'}(t+1) = w \times V_i(t) + C_1 \times rand_1 \times [G_i(t) - X_i(\mathrm{t})] + C_2 \times rand_2 \times [P_g(t) - X_i(\mathrm{t})] \tag{8}$$

$$V_i^{'}(t+1) = w \times V_i(t) + C_1 \times rand_1 \times [P_i(t) - X_i(\mathrm{t})] + C_2 \times rand_2 \times [G_i(t) - X_i(\mathrm{t})] \tag{9}$$

$$V_i^{'}(t+1) = w \times V_i(t) + C_1 \times rand_1 \times [G_i(t) - X_i(\mathrm{t})] + C_2 \times rand_2 \times [G_i(t) - X_i(\mathrm{t})] \tag{10}$$

The results presented indicate that only the method in (8) performs better than the standard PSO of (1). This improvement comes at increased processing cost, as the clustering algorithm needs a nonnegligible amount of time to form the stereotypical groups. In a time-varying system we define:

$$G_i(t) = \frac{1}{h}\sum_{l=0}^{h} P_i(t-l) \tag{11}$$

Where h is a width of window.

Moreover, following simulations have shown. We are able to make use of the advantage of more approaches in the time-varying system. For example we take two approaches of PSO. We can divide particles of swarm into two types crossly. The first type is used for FSPSO. Another type is used for the PSO of inertia weights in (1). The particles of swarm are two times more than number of parameters in identification. The particles of each type are more than parameters in identification.

3. IDENTIFICATION FOR ARMAX MODEL WITH TIME-VARYING PARAMETERS

The considered stochastic ARMAX model with time-varying parameters is given by[15]

$$A(q^{-1},k)y(k) = q^{-d(k)}B(q^{-1},k)u(k) + \frac{1}{C(q^{-1},k)}e(k) \tag{12}$$

where

$$A(q^{-1},k)=1+a_1(k)q^{-1}+a_2(k)q^{-2}+\Lambda\ +a_{na}(k)q^{-na},$$
$$B(q^{-1},k)=b_1(k)q^{-1}+b_2(k)q^{-2}+\Lambda\ +b_{nb}(k)q^{-nb},$$
$$C(q^{-1},k)=1+c_1(k)q^{-1}+c_2(k)q^{-2}+\Lambda\ +c_{nc}(k)q^{-nc}.$$

$a_1(k), a_2(k), \Lambda\ , a_{na}(k), b_1(k), b_2(k), \Lambda\ , b_{nb}(k), c_1(k), c_2(k), \Lambda\ , c_{nc}(k)$, are the time-varying parameters of the system. $d_{\min} \le d(k) \le d_{\max}$ is the time-varying delay. *u (k)*, *y(k)* and *e(k)* are system input, output and white noise serial respectively. If $C(q^{-1},k)\neq 1, \dfrac{e(k)}{C(q^{-1},k)}$ shows a type of colored noise in (12). Suppose the model of the system is

$$\hat{A}(q^{-1},k)\hat{y}(k)=q^{-\hat{d}(k)}\hat{B}(q^{-1},k)u(k)+\frac{1}{\hat{C}(q^{-1},k)}e(k) \tag{13}$$

where

$$\hat{A}(q^{-1},k)=1+\hat{a}_1(k)q^{-1}+\hat{a}_2(k)q^{-2}+\Lambda\ +\hat{a}_n(k)q^{-na},$$
$$\hat{B}(q^{-1},k)=\hat{b}_1(k)q^{-1}+\hat{b}_2(k)q^{-2}+\Lambda\ +\hat{b}_{nb}(k)q^{-nb},$$
$$\hat{C}(q^{-1},k)=1+\hat{c}_1(k)q^{-1}+\hat{c}_2(k)q^{-2}+\Lambda\ +\hat{c}_{nc}(k)q^{-nc}$$

$\hat{\theta}^T(k)=[\hat{d}(k),\hat{a}_1(k),\hat{a}_2(k),\Lambda\ ,\hat{a}_{na}(k),\hat{b}_1(k),\hat{b}_2(k),\Lambda\ ,\hat{b}_{nb}(k),\hat{c}_1(k),\Lambda\ ,\hat{c}_{nc}(k)]$ is $\theta^T(k)=[d(k),a_1(k),a_2(k),\Lambda\ ,a_{na}(k),b_1(k),b_2(k),\Lambda\ ,b_{nb}(k),c_1(k),\Lambda\ ,c_{nc}(k)]$ estimation of the parameters at *k* time. The error between the actual and the estimated system output is defined by

$$\begin{aligned}\varepsilon = y(k)-\hat{y}(k) = &[\frac{B(q^{-1},k)}{A(q^{-1},k)}q^{-d(k)}-\frac{\hat{B}(q^{-1},k)}{\hat{A}(q^{-1},k)}q^{-\hat{d}(k)}]u(k)\\ &+[\frac{1}{A(q^{-1},k)C(q^{-1},k)}-\frac{1}{\hat{A}(q^{-1},k)\hat{C}(q^{-1},k)}]e(k)\end{aligned} \tag{14}$$

When the identification model is different from the actual system, $\varepsilon \neq 0$. We define the performance criterion function is as follows

$$\begin{aligned}J_h(\theta(k)) = \sum_{i=0}^{h}\lambda^i\{&[y(k-i)-\hat{y}(k-i)]^2+\\ &\mu[\theta_{best}(k-i)-\hat{\theta}(k-i)]^T[\theta_{best}(k-i)-\hat{\theta}(k-i)]\}\end{aligned} \tag{15}$$

Where h is a width of window. The faster parameters of time-varying change, the smaller choice h is to have batter result. $\overset{\wedge}{y(k)}$ is the estimated output in system. λ is the forgetting factor. Typically, $0<\lambda\le 1$ is the range of λ .Actually we use a value of λ from 0.90 to 0.98. The much smaller λ^i is, the more *i* increase in order to track the dynamic system and forget older data. μ shows coefficient in square error of parameters. Its value may balance the ratio of error of parameters and system output.

Actually we use a value of μ from 0.3 to 0.5. In (15) the first part shows error of system output and second part shows error of parameters in order to explain more difference between actual and estimative system.

A flow chart for such an algorithm, referred to here as combinatorial PSO, is given in Figure 1. This algorithm is capable of providing desirable performance and convergence properties in most any context.

In (12) all parameters of model are given: a_1 (k) =-1.5, a_2 (k) =0.7, b_2 (k) =0.5, c_1 (k) =1.0, c_2 (k) =0.41, d (k) =2. Where the time-varying parameter is

$$b_1(k) = \begin{cases} 1.0 & (k < 200) \\ 1 + 0.4 * \sin [0.2(k - 200)] & (k \geq 200) \end{cases}$$

The noise e(k) is the white noise whose mean is null and $\sigma^2 = 0.1$. The input signal $u(k)$ is the white noise whose mean is null and amplitude is 1.

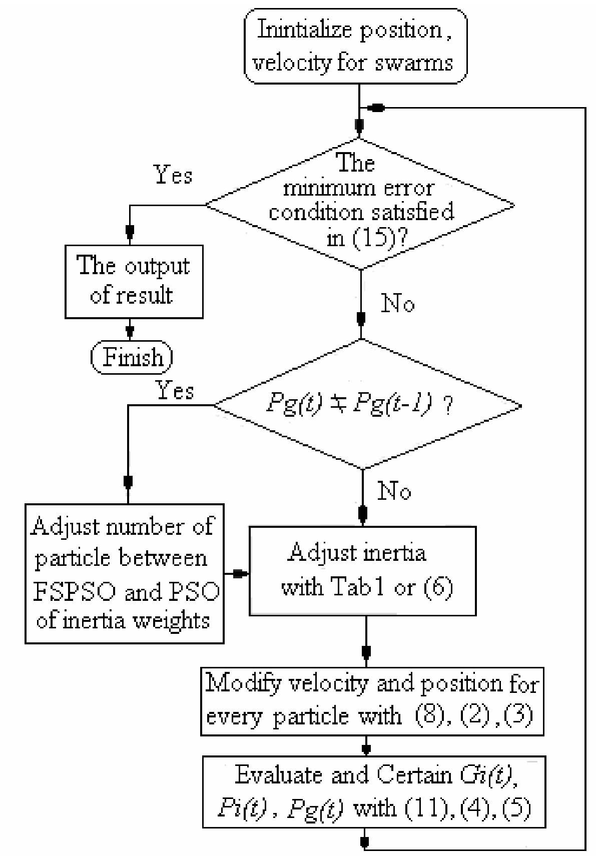
Inintialize position, velocity for swarms
The minimum error condition satisfied in (15)?
Yes
The output of result
Finish
No
Pg(t) ≒ Pg(t-1) ?
Yes
No
Adjust number of particle between FSPSO and PSO of inertia weights
Adjust inertia with Tab1 or (6)
Modify velocity and position for every particle with (8), (2), (3)
Evaluate and Certain Gi(t), Pi(t), Pg(t) with (11), (4), (5)

Figure 1 - A flow chart for such an algorithm based on combinatorial PSO

In following figures horizontal coordinate is iteration times and vertical coordinate is value of parameters. In figures green line is actual value of parameters and red stippling is estimated value of parameters. In order to show clearly in figures we only give out the result of a_1 *(k)* and b_1 *(k)*. a_1 *(k)* represents time invariant parameter. b_1 *(k)* represents time-varying parameter. In order to avoid bad convenience and velocity too fast to control range of velocity we set the maximum velocity into 1 ($V_{d\max} = 1$). The number of particles N_1 takes 30. Acceleration coefficients C_1 and C_2 take 2 equally. The forgetting factor λ takes 0.95. μ takes 0.4.

4. DIGITAL SIMULATION

4.1 Identification with PSO of Inertia Weights

Figure 2 shows the result in the PSO of inertia weights when h is 3. Its inertia weights changes from 0.9 to 0.4 with (6). Similarly Figure 3 and Figure 4 are results when h is 5 and 10 respectively. From results we may find that the better result of tracking parameter is, the smaller h is. The later tracking of parameters is batter than forward convenience in PSO of inertia weights. Before 200 iteration times result is not good.

4.2 Identification with FSPSO

Figure5 shows the result in FSPSO when h is 3. Its inertia weights changes from 0.9 to 0.4 with fuzzy logic rules in Table 1. Similarly Figure 6 and Figure 7 are results of h=5 and 10 respectively. From results we may find that forward convenience of parameters is batter than the later tracking of parameters in FSPSO. Its convenience (about 70 iteration times) is faster than the PSO of inertia weights (about 200 iteration times). The result with h=3 (Figure 5) is batter than other (Figure 6 or Figure 7).

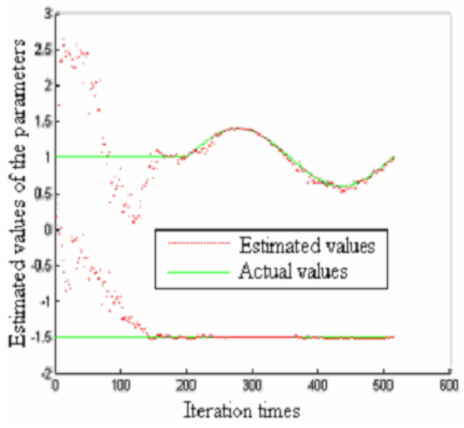

Figure 2 - PSO of inertia weights with h=3

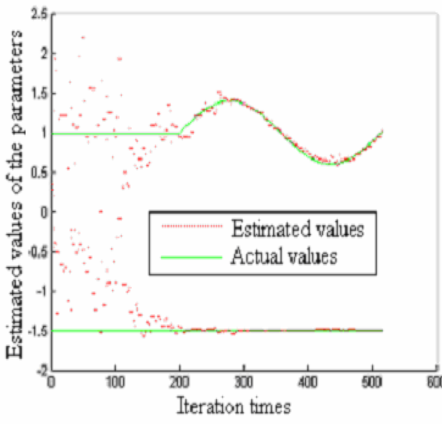

Figure3 - PSO of inertia weights with h=5

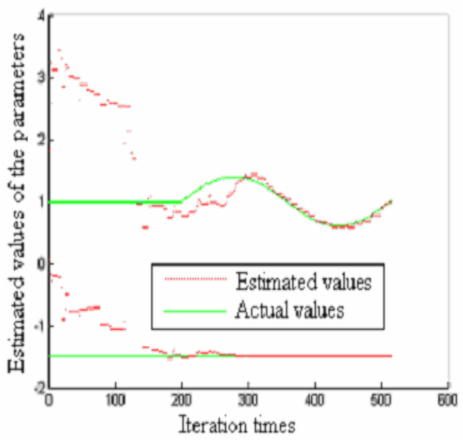

Figure 4 - PSO of inertia weights with h=10

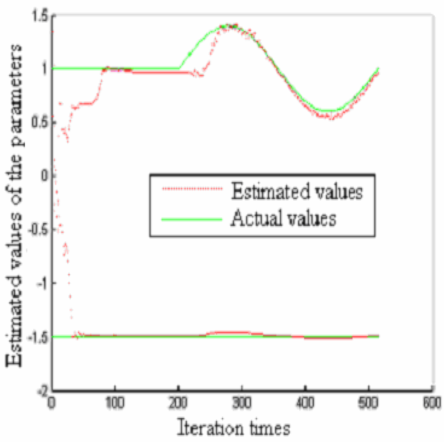

Figure 5 - FSPSO with h=3

We may compare FSPSO and the PSO of inertia weights with above results. Then we can discover that FSPSO combines the combinatorial PSO with the PSO of inertia a weights to take thire advantages and to make up another degradation.

4.3 Identification with combinatorial PSO

If we use combinatorial PSO we meet the allocation of swarms. According to lots of simulation we select an half particles (15 particles) for FSPSO and other particles (15 particles) for PSO of inertia weights. Figure 8 shows the result in combinatorial PSO when h is 3. Similarly Figure 9 and Figure 10 are results when h is 5 and 10 respectively. It is seen that in the combinatorial PSO tracking of parameters and convenience is much batter than that one in FSPSO or the PSO of inertia weights. The better result of tracking parameter is, the smaller h is when h is more than 2. Or the convenience is imperfect.

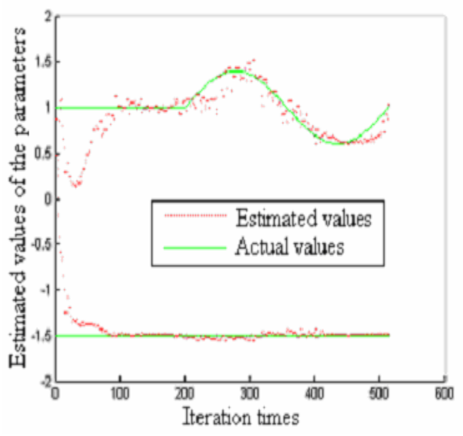

Figure 6 - FSPSO with h=5

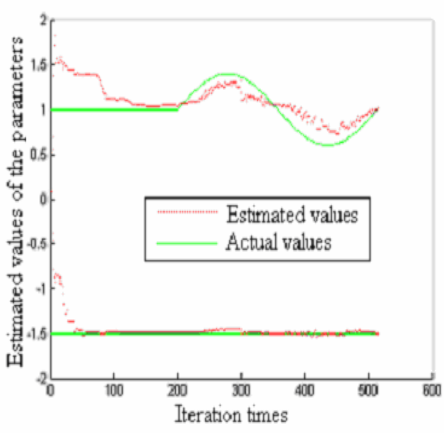

Figure 7 - FSPSO with h=10

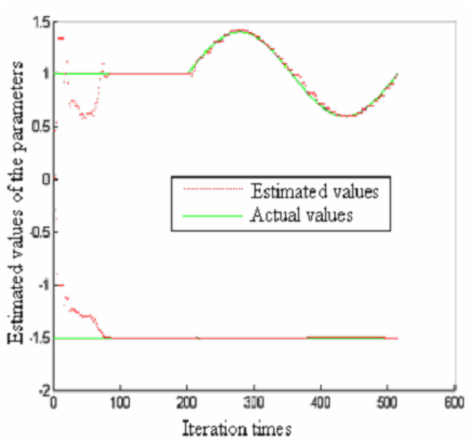

Figure 8 - Combinatorial PSO with h=3

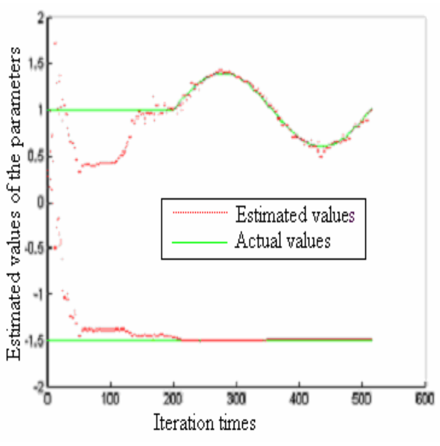

Figure 9 - Combinatorial PSO with h=5

A lots of simulations show that the more particles for the PSO of inertia weights are, the batter dynamic tracing is and that the more particles for FSPSO are the batter the convergence of time invariant systems is.

4.4 The robustness with colored noise in system

In the above results h=3 is good selection. When the ratio of colored noise to signal is 10 percentages simulation result is shown in Figure 11. Similarly Figure 12 and Figure 13 are shown respectively when the ratios of colored noise to signal are 20 and 30 percentages. Results of tracking and robustness are more satisfied.

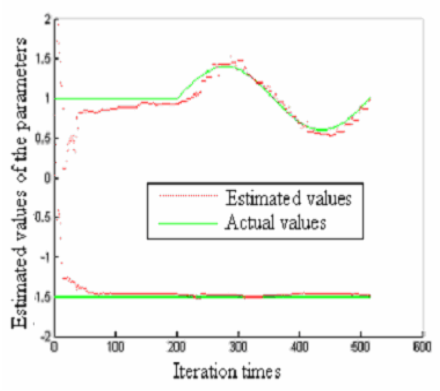

Figure 10 - Combinatorial PSO with h=10

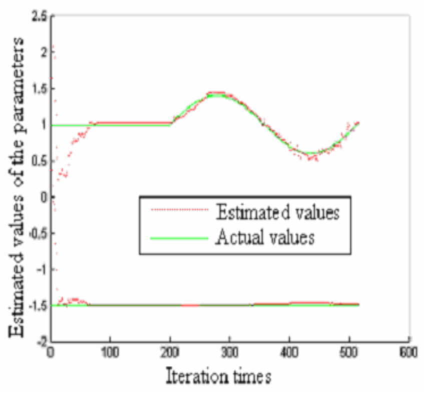

Figure 11 - Combinatorial PSO with noise 10%

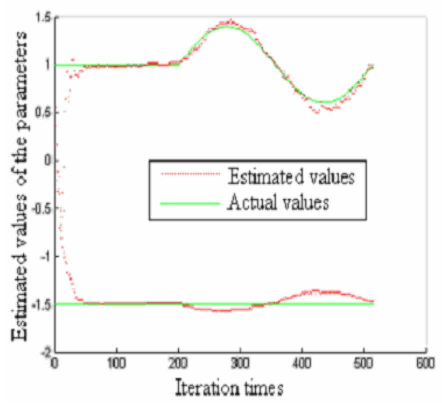

Figure 12 - Combinatorial PSO with noise 20%

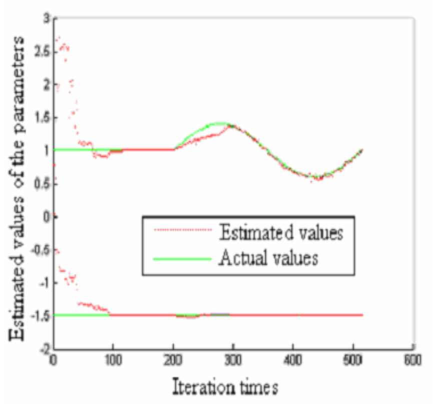

Figure 13 - Combinatorial PSO with noise 30%

5. CONCLUSIONS

In this paper, parameter estimation of the time varying for process models is converted to an optimization problem. Presented (15) can show not only outside system output error but also inside parameters error in order to explain more difference between actual and estimative system. We are able to make use of advantages of more approaches in the time-varying system. We take the combinatorial PSO that FSPSO combines PSO of inertia weights in simulation.

The identification algorithm for time-varying systems with colored noise was indeed more efficient and robust in combinatorial PSO comparing with FSPSO or PSO of inertia weights.

6. ACKNOWLEDGEMENT

This work is supported by the K. C. Wong Education Foundation

7. REFERENCES

1. Feng Ding, Tongwen Chen, "Performance Bounds of Forgetting Factor Least-Squares Algorithms for Time-Varying Systems with Finite Measurement Data", *IEEE Trans. Circuits and Systems,* 52(3): 555-566, 2005.
2. R. Salomon, "Evolutionary algorithms and gradient search similarities and differences," *IEEE Trans. Evolutionary Computation*: 2(2), 45-55, 1998.
3. Yuncan Xue, Qiwen Yang, Jixin Qian, "Parameter estimation for time-varying system based on improved genetic algorithm", *Proc. the 28 Annual Conference of the IEEE Industrial Electronics Society*, Sevilla, Spain: 2007-2010, 2002.
4. R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proc. 6th Industrial Symp. Micro Machine and Human Science*, Nagoya, Japan: 39–43, 1995.
5. P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," *Proc. Conference Evolutionary computation*, Washington, DC: 1958–1961, 1999.

6. R. C. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*: Academic, ch. 6: 212–226, 1996.
7. Weixing Lin, Chongguang Jiang, Jixin Qian, “The Identification of Hammerstein Model Based on PSO with Fuzzy Adaptive Inertia Weight”, *Journal of Systems Science and Information*, 3(2): 381-391, 2005.
8. Y. Shi, R. C. Eberhart, “Empirical Study of particle swarm optimization”, *Proc. IEEE International Conference. Evolutionary Computation*, 3: 101-106, 1999.
9. M. Clerc, “The swarm and the queen: toward a deterministic and adaptive particle swarm optimization,” *Proc. ICEC’99*, Washington, DC: 1951–1957, 1999.
10. D. Corne, M. Dorigo, F. Glover, Eds., *New Ideas in Optimization*. New York: McGraw-Hill, ch. 25: 379–387, 1999.
11. M. Clerc, J. Kennedy, “The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space,” *IEEE Trans. Evolutionary Computation*, 6(1): 58–73, 2002.
12. R. C. Eberhart, Y. Shi, “Comparing inertia weights and constriction factors in particle swarm optimization”, *Proc. Conference Evolutionary Computation 2000*, San Diego, CA: 84-88, 2000.
13. [13] J. Kennedy, “Stereotyping: Improving particle swarm performance with cluster analysis,” *Proc. 2000 Conference Evolutionary Computing*: 1507–1512, 2000.
14. J. Kennedy, R. Mendes, “Population structure and particle swarm performance”, *Proc. 2002 World Conference Computational Intelligence*, Honolulu, HI: 1671–1676, 2002.
15. Jing Ke, Yizheng Qiao, Jixin Qian, “ Identification of Time-varying Delay Systems Using Particle Swarm Optimization”, *Proc. the 5th World Congress on Intelligent Control and Automation*, Hangzhou, P.R. China: 330-334, 2004.