J. Software Engineering & Applications, 2010, 3: 221-229
doi:10.4236/jsea.2010.33027 Published Online March 2010 (http://www.SciRP.org/journal/jsea)

Parameter Identification Based on a Modified PSO Applied to Suspension System

Alireza Alfi, Mohammad-Mehdi Fateh

Faculty of Electrical and Robotic Engineering, Shahrood University of Technology, Shahrood, Iran.
Email: a_alfi@shahroodut.ac.ir

Received November 25th, 2009; revised January 15th, 2010; accepted January 20th, 2010.

ABSTRACT

This paper presents a novel modified particle swarm optimization algorithm (*MPSO*) *for both offline and online parametric identification of dynamic models. The MPSO is applied for identifying a suspension system introduced by a quarter-car model. A novel mutation mechanism is employed in MPSO to enhance global search ability and increase convergence speed of basic PSO* (*BPSO*) *algorithm. MPSO optimization is used to find the optimum values of parameters by minimizing the sum of squares error. The performance of the MPSO is compared with other optimization methods including BPSO and Genetic Algorithm* (*GA*) *in offline parameter identification. The simulating results show that this algorithm not only has advantage of convergence property over BPSO and GA, but also can avoid the premature convergence problem effectively. The MPSO algorithm is also improved to detect and determine the variation of parameters. This novel algorithm is successfully applied for online parameter identification of suspension system.*

Keywords: *Particle Swarm Optimization*, *Genetic Algorithm*, *Parameter Identification*, *Suspension System*

1. Introduction

A mathematical model can be provided to describe the behavior of a system based on obtained data for its inputs and outputs by system identification. It is necessary to use an estimated model for describing the relationships among the system variables for this purpose. The values of parameters in the estimated model of a system must be found such that the predicted dynamic response coincides with that of the real system [1].

The basic idea of parameter identification is to compare the time dependent responses of the system and parameterized model based on a performance function giving a measure of how well the model response fits the system response. It should be mentioned that the model of system must be regarded fixed through identification procedure. It means that data are collected from the process under a determined experimental condition and after that, the characteristic property of system will stay the same.

Many traditional techniques for parameter identification have been studied such as the recursive least square [2], recursive prediction error [3], maximum likelihood [4], and orthogonal least square estimation [5]. Despite their success in system identification, traditional optimization techniques have some fundamental problems including their dependence on unrealistic assumptions such as unimodal performance landscapes and differentiability of the performance function, and trapping in local minima [6].

Evolutionary algorithms (EAs) and swarm intelligence (SI) techniques seem to be promising alternatives as compared with traditional techniques. First, they do not rely on any assumptions such as differentiability, continuity, or unimodality. Second, they can escape from local minima. Because of this, they have shown superior performances in numerous real-world applications. Among them, genetic algorithm (GA) and particle swarm optimization (PSO) are frequently used algorithms in the area of EAs and SI, respectively. Owing these attractive features, these algorithms are applied in the area of system identification [7–10].

Comparing GA and PSO, both are population based optimization tools. However, unlike GA, PSO has no evolution operators such as crossover and mutation. Easy to implement and the less computational complexity are advantages of PSO in comparing with GA. The basic PSO (BPSO) algorithm has good performance when dealing with some simple benchmark functions. However, it is difficult for BPSO algorithm to overcome local minima when handling some complex or multimode functions. Hence, a modified PSO (MPSO) is proposed

to overcome this shortage. In this paper, a novel mutation mechanism is introduced to enhance global search of algorithm. Then, it is demonstrated how to employ the MPSO method to obtain the optimal parameters of a dynamic system.

In order to show the effectiveness of MPSO in system identification, a quarter-car model of suspension system is identified as an application. Although a linear model is proposed for a suspension system for control purposes [11–14], the MPSO can be applied well to identify the non-linear systems, as well. It should be noticed that a suspension system operates under various operating conditions, where parameter variations are unavoidable. Accurate knowledge of these parameters is important to form the control laws. Therefore, it is of our interest to investigate an efficient model parameter tracking approach to achieve precise modeling results under different conditions without using complicated model structures.

In this paper, the MPSO is compared to GA and BPSO in offline parameter identification of suspension system. It can be shown that the MPSO has a better performance than the aforementioned algorithms in solving the parameter estimation of suspension system. Because of the superiority of MPSO in offline identification, it can be used for online parameter identification of suspension system, as well. In the propose method, the estimated parameters will not be updated unless any changes in system parameters is detected by algorithm. A sentry particle is introduced to detect any change in system parameters. If a change is detected, the algorithm scatters the particles around the global best position and forces the algorithm to forget its global memory, then runs the MPSO to find the new values for parameters. Therefore, MPSO runs further iterations if any changes in parameters are detected.

The rest of the paper is organized as follow: Next section describes problem description. Section 3 introduces optimization algorithms. The proposed algorithms in both offline and online parametric identification are presented in Section 4. Simulation results are shown in Section 5. Finally, conclusion and future works are presented in Section 6.

2. Problem Description

This section presents a quarter-car model of suspension system and a proper fitness function for optimization algorithms.

2.1 Suspension System Dynamics

Modeling of vehicle suspension system has been studiedfor many years. In order to simplify the model, a quarter-car model was introduced in response the vertical force for the suspension system [15] as shown in **Figure 1**. In this figure, b is damping coefficient, m_1 and m_2 are unsprung and sprung mass, respectively, k_1 and k_2 are tire and suspension stiffness, respectively, u is the road displacement and y is the vertical displacement of sprung mass. The linearized dynamic equations at equilibrium point with an assumption that the tire is in contact with the road are given as:

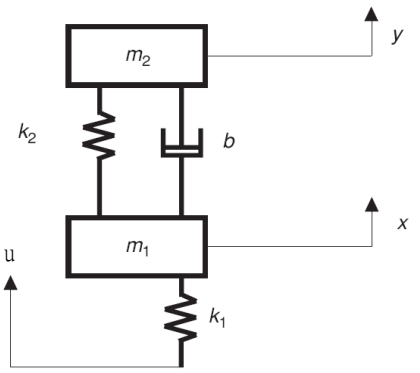


Figure 1. Schematic diagram of the quarter-car model

$$m_1 \frac{d^2x}{dt^2} = k_2(y-x) + b\left(\frac{dy}{dt} - \frac{dx}{dt}\right) + k_1(u-x) \quad (1)$$

$$m_2 \frac{d^2y}{dt^2} = -k_2(y-x) - b\left(\frac{dy}{dt} - \frac{dx}{dt}\right) \quad (2)$$

2.2 Problem Statement

When the model of system is fixed through identification procedure, the parameter identification problem can be treated as an optimization problem. The basic idea of parameter estimation is to compare the system responses with the parameterized model based on a performance function giving a measure of how well the model response fits the system response. Moreover, a common rule in identification is to use excitation signals that correspond to a realistic excitation of the system such that the identified linear model is a good approximation of the system for that type of excitation. Consequently, in order to estimate the system parameters, excitation signal is chosen Gaussian band-limited white noise. The bandwidth is set to 50 Hz, which is sufficiently higher than the desired closed-loop bandwidth [14].

Considering **Figure 2** the excitation input is given to both the real system and the estimated model. Then, the outputs from the real system and its estimated model are input to the fitness evaluator, where the fitness will be calculated. The sum of squares error between real and estimated responses for a number of given samples is considered as fitness of estimated model. So, the fitness function is defined as follow:

$$\mathrm{SSE} = \sum_{k=1}^{N} e^2 = \sum_{k=1}^{N} \left(y(kT_s) - \hat{y}(kT_s)\right)^2 \quad (3)$$

where N is the number of given sampling steps, $y(kT_s)$

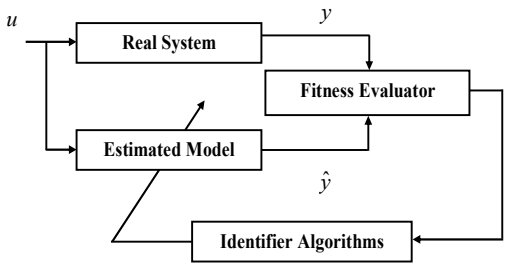


Figure 2. The estimation process

and $\hat{y}(kT_s)$ are real and estimated values in each sample time, respectively. The calculated fitness is then input to the identifier algorithms, *i.e.* GA- BPSO and MPSO, to identify the best parameters for estimated system in fitting procedure by minimizing the sum of square of residual errors in response to excitation input.

3. Optimization Algorithms

As mentioned before, the parameter identification problem can be treated as an optimization problem. The proposed MPSO optimization algorithm is compared with frequently used algorithms in optimization problems, namely GA and BPSO in the optimization problem in hand. These algorithms are taken from two main optimization groups namely evolutionary algorithms (EAs) and swarm intelligence (SI). These algorithms are currently used for numerical optimization problems of stochastic search algorithms.

3.1 Evolutionary Algorithms (EAs)

EAs algorithms are population based, instead of using a single solution. EAs mimic the metaphor of natural biological evolution. EAs operate on a population of potential solutions applying the principle of survival of the fittest to produce better approximations to a solution. At each generation, a new set of approximations is created by two processes. First, selecting individuals according to their level of fitness in the problem domain. Second, breeding them together using operators borrowed from natural genetics.

This process leads to the evolution of populations of individuals that are better suited to their environment than they were created from, just as in natural adaptation. Evolutionary algorithms model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. The majority of the present implementations of EA come from any of these three basic types, which are strongly related although independently developed: Genetic Algorithms (GA), Evolutionary Programming (EP) and Evolutionary Strategies (ES). Hence, in this paper, the proposed method is compared to GA.

3.2 Swarm Intelligence (SI)

SI is the artificial intelligence based on the collective behavior of decentralized and self-organized systems. SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules. Although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling. Among them, PSO is a new and frequently used SI technique.

3.2.1 Basic PSO

PSO is used to search for the best solution by simulating the movement and flocking of birds [16]. The algorithm works by initializing a flock of birds randomly over the searching space, where every bird is called as a "particle". These "particles" fly with a certain velocity and find the global best position after some iteration. At each iteration, each particle can adjust its velocity vector based on its momentum and the influence of its best position as well as the best position of the best individual. Then, the particle flies to a new computed position. Suppose that the search space is n-dimensional, and then the position and velocity of particle i are represented by $X_i = [x_{i1},\ x_{i2},\ \ldots\ , x_{in}]^T$ and $V_i = [v_{i1},\ v_{i2},\ \ldots\ , v_{in}]^T$, respectively. The fitness of each particle can be evaluated according to the objective function of optimization problem. The best previously visited position of the particle i is noted as its personal best position denoted by $P_i = [p_{i1},\ p_{i2},\ \ldots\ , p_{in}]^T$. The position of the best individual of the swarm is noted as the global optimum position $G = [g_1,\ g_2,\ \ldots\ , g_n]^T$. At each step, the velocity of particle and its new position will be assigned as follows:

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 (P_i - X_i) + c_2 r_2 (G - X_i) \quad (4)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (5)$$

where ω is called the inertia weight that controls the impact of previous velocity of particle on its current one. r_1 and r_2 are independently uniformly distributed random variables in a range of [0,1]. c_1 and c_2 are positive constant parameters called acceleration coefficients which control the maximum step size. In the references [17,18], several strategies of inertial weight ω were given. Generally, the inertial weight ω should be reduced rapidly in the beginning stages of algorithm but it should be reduced slowly around optimum. If the velocity exceeds the predefined limit, another restriction called $V_{\max}$ is used.

In BPSO, (4) is used to calculate the new velocity ac-

cording to its previous velocity, the distance of its current position from both its own personal best position and the global best position of the entire population. Then the particle flies toward a new position according (5). This process is repeated until a stopping criterion is reached.

3.2.2 The Proposed Modified PSO

As mentioned before, possible trapping in local minima when handling some complex or multimode functions is a shortage of BPSO [19–21]. Hence, the motivation of the proposed method is to overcome this drawback. In BPSO, as time goes on, some particles become quickly inactive because their states are similar to the global optimum. As a result, they lose their velocities. In the subsequent generations, they will have less contribution to the search task due to their very low global search activity. In turn, this will induce the emergence of a state of premature convergence.

To deal with the problem of premature convergence, several investigations have been undertaken to avoid the premature convergence. Among them, many approaches and strategies are attempted to improve the performance of PSO by variable parameters. A linearly decreasing weight into PSO was introduced to balance the global exploration and the local exploitation in [17]. PSO was further developed with time-varying acceleration coefficients to modify the local and the global search ability [18]. A modified particle swarm optimizer with dynamic adaptation of inertia weight was presented [19]. Moreover, some approaches aim to divert particles in swarm among the iterations in algorithm. Mutation PSO employed to improve performance of PSO [20]. The concepts of ‘‘subpopulation” and ‘‘breeding” adopted to increase the diversity [21]. An attractive and repulsive PSO developed to increase the diversity [22].

In this paper, a modified particle swarm optimization (MPSO) algorithm is proposed to avoid premature convergence and increase the convergence speed of algorithm. In our proposed method, after some iteration, the algorithm measures the search ability of all particles and mutates a percentage of particles which their search ability is lower than the others. Our motivation is that particles with low search ability become inactive as their fitness do not grow and need to mutate for getting a chance to search new areas in solution space, which may not been meet already. Also the mutation rate is not constant and if the global best doesn't grow, the rate of mutation is increased. If the fitness of global optimum does not grow, the algorithm can get stuck in local minima forever or at least for some iteration, which lead to a slow convergence speed. In the other words, if the global best of the present population is equal to that of the previous population (solution converges), mutation rate is set to a higher value P_{mh}, such that the diversity of particles is increased so to avoid premature convergence. Otherwise (solution diverges), P_m is set to a lower value P_{ml}, since the population already has enough diversity. The adaptive mutation rate scheme is described as

$$P_m = \begin{cases} P_{mh} & ;\text{if G(t)=G(t-1)} \\ P_{ml} & ;\text{if G(t)>G(t-1)} \end{cases} \tag{6}$$

where P_{mh} and P_{ml} are 0.2 and 0.1, respectively. Consequently the mutation rate will be increased by 0.1, if the global optimum doesn't grow, until a growth on the fitness of global optimum occurs. In order to measure the search ability for particle i at each iteration, we take advantage of the fitness increment of the local optima in a designated interval ∇T, from iteration $t-\nabla T$ to t. consequently; the parameter C_i is used to measure the search capability of particle i.

$$C_i = \left| F(P_i(t)) - F(P_i(t-\nabla T)) \right| \tag{7}$$

where $F(P_i(t))$ is the fitness of the best position for i-th particle in t-th iteration $P_i(t)$, and ∇T is a designated interval. Particles with low search ability (small C_i) have low increment in the best local value for their low global search activity and so in our algorithm, they have a bigger change to mutate to get a chance to search a new area in the search space.

Generally, In MPSO algorithm, after iteration ∇T, all particles are sorted according to their C parameter. Then the swarm is divided into two parts: The active part including the top $(1-P_m)\times S$ with higher C and the inactive part consisting of the rest $P_m \times S$ particles with smaller C whereas S is size of the swarm. Particles in the first part update their velocities and position the same as BPSO algorithm. Finally, in order to increase the diversity of the swarm, the inactive particles are chosen to mutate by adding a Gauss random disturbance to them as follows:

$$x_{ij} = x_{ij} + \beta_{ij} \tag{8}$$

where x_{ij} is the j-th component of the i-th inactive particle. β_{ij} is a random variable, which follows a Gaussian distribution with a mean value of zero and a variance value of 1, namely $\beta_{ij} = N(0,1)$. This strategy prevents all particles to divert from the local convergence. Instead, only inactive particles are mutated.

4. Implementation of the MPSO

In this section, the procedure of MPSO in online and offline system parameter identification is described.

4.1 Offline Identification

MPSO algorithm is applied to find the best system parameter, which simulates the behavior of dynamic system. Each particle represents all parameters of estimated model. The procedure for this algorithm can be summarized as follows:

Step 1: Initialize positions and velocities of a group of particles in an M-dimensional space as random points where M denotes the number of system parameters;

Step 2: Evaluate each initialized particle's fitness value using (3);

Step 3: Set P_i as the positions of current particles while G is the global best position of initialized particles. The best particle of current particles is stored;

Step 4: The positions and velocities of all particles are updated according to (4) and (5), and then a group of new particles are generated;

Step 5: Evaluate each new particle's fitness value. If the new position of *i*-th particle is better than P_i, set P_i as the new position of the *i*-th particle. If the fitness of best position of all new particles is better than fitness of G, then G is updated and stored;

Step 6: If iteration $> \Delta T$, calculate the mutation rate (P_m) and search ability of each particle using (6) and (7), respectively. Then mutate $P_m \times S$ number of particle with lower search ability;

Step 7: Update the velocity and location of each particle according to the (4) and (5). If a new velocity is beyond the boundary $[V_{\min}, V_{\max}]$, the new velocity will be set as $V_{\min}$ or $V_{\max}$;

Step 8: Output the global optimum if a stopping criteria is achieved, else go to Step 5.

When a stopping criterion is occurred, the global optimum is the best answer for the problem in hand (the best estimated system parameters).

4.2 Online Identification

The proposed algorithm sequentially gives a data set by sampling periodically. The optimized values of parameters for the first data set are determined by using a procedure described in Subsection 4.1. The estimated parameters will not be updated unless a change in the system parameters is detected. In order to detect any change in system parameters, the global optimum in the later period is noticed as a sentry particle. In each period, the sentry particle is evaluated at first and if the fitness of the sentry particle in the current period is bigger than the previous one, the changes in parameters are confirmed. If no changes are detected, the algorithm leaves this period without changing the positions of particles. When any changes in parameters occur, the algorithm runs further to find the new optimum values. For this purpose, a new coefficient (Δ) is introduced as follows:

$$\Delta = \frac{fitness(\mathrm{S_p}(i)) - fitness(\mathrm{S_p}(i-1))}{fitness(\mathrm{S_p}(i))} \quad (0 \le \Delta < 1) \tag{9}$$

where $\mathrm{S_p}(i)$ is the sentry particle in the *i*-th period. Δ will be bigger than zero if the fitness of the sentry particle at the current period is bigger than the previous one. Thus, changes in model parameters are detected by inspecting Δ at each period. In this case, the particles in population must forget their current global and personal memories in order to find the new global optimum. The fitness of global optimum particle and personal bests of all particles are then evaporated at the rate of a big evaporation constant. As a result, other particles have a chance of fitness bigger than the previous global optimum. Moreover, the velocities of particles are increased to search in a bigger solution space for new optimal solution. When a change in system parameters is detected, the following changes are considered.

$$fitness(P_i) = fitness(P_i) \times T \quad i = 1,...,S \tag{10}$$

$$fitness(\mathrm{G})_{new} = fitness(\mathrm{G})_{old} \times T \tag{11}$$

$$V_{new} = V_{old} + \Delta \times V_{\max} \tag{12}$$

T is an evaporation constant. Also (12) shows that the velocity of particles increase by $\Delta \times V_{\max}$ only in one iteration. Notice that a bigger Δ, *i.e.* greater changes in parameters, causes a bigger velocity. This means that if significant changes in system parameters occur, the particles must search a bigger space and if a little change occurs, particle search around the previous position to find the new position. This strategy accelerates convergence speed of the algorithm, which is an important issue in online identification.

5. Simulation Results

In this section the proposed MPSO algorithm is applied to identify parameters of a suspension system, which its nominal parameters are summarized in **Table 1** [15]. In order to show the performance of the proposed MPSO in

Table 1. Suspension parameters [15]

Parameters	Nominal value
m_1	26 kg
m_2	253 kg
k_1	90000 N/m
k_2	12000 N/m
b	1500 N·sec/m

the problem in hand, it is compared to two frequently used optimization algorithms, including GA and BPSO. Simulation results have been carried out in two parts.

In the first part, in order to show the effectiveness of the proposed MPSO in offline identification, it has been compared with GA and BPSO. In the second part, the proposed MPSO is applied to online parameter identification for suspension system. In both BPSO and MPSO algorithms, $c_1 = c_2 = 2$, and the inertia weight is set to 0.8. Also, the simulation results are compared with GA, where the crossover probability P_c and the mutation probability P_m are set to 0.8 and 0.1, respectively.

5.1 Offline Parameter Identification

Owing to the randomness of the heuristic algorithms, their performance cannot be judged by the result of a single run. Many trials with different initializations should be made to acquire a useful conclusion about the performance of algorithms. An algorithm is robust if it gives consistent result during all the trials. The searching ranges are set as follows:

$20 \le m_1 \le 30$, $200 \le m_2 \le 300$, $85000 \le k_1 \le 90000$, $10000 \le k_2 \le 15000$, $1200 \le b \le 1700$.

In order to run BPSO, MPSO and GA algorithms, a population with a size of 10 for 100 iterations is used.

Comparison of results on sum of squares error resulted from 20 independent trials with N = 1000, 2000 and 2500 are shown in **Tables 2–4**, respectively. This comparison shows that the MPSO is superior to GA and BPSO. Moreover, MPSO is significantly more robust than other algorithms because the best and the mean values obtained by MPSO are very close to the worst value. In addition, the convergence speed of GA, BPSO and MPSO are compared. **Figure 3** shows the convergence speed of these algorithms during 100 iterations which proves that the convergence speed of the proposed MPSO is faster than GA and BSO which can be conclude that MPSO is more proper than aforementioned algorithms. **Figure 4** confirms the success of optimization process by using MPSO algorithm. The identified parameters are m_1, m_2, k_1, k_2 and b, respectively. In this figure, the data set is formed by 1000 samples. In addition, to compare computational time of these algorithms, a threshold of 10^{-5} is considered as stopping condition, in contrast to a predefined number of generation. Then each algorithm runs 20 times and the average of elapsed time is considered as a criteria for computational time. **Table 5** illustrates the results obtained by GA, BPSO, and MPSO. It is clearly obvious that, the proposed algorithm spends extremely fewer iteration and less computational time to reach a predefined threshold as compared with other algorithms. Hence, it can be concluded that IPSO is more proper than

Table 2. Comparison of GA, BPSO and MPSO in offline identification for N=1000

	SSE		
	Best	Mean	Worst
GA	3.11×10^{-6}	2.11×10^{-4}	3.1117×10^{-3}
BPSO	6.45×10^{-8}	1.78×10^{-6}	2.18×10^{-5}
MPSO	3.12×10^{-10}	1.64×10^{-9}	8.46×10^{-9}

Table 3. Comparison of GA, BPSO and MPSO in offline identification for N = 2000

	SSE		
	Best	Mean	Worst
GA	6.98×10^{-6}	1.24×10^{-3}	4.65×10^{-3}
BPSO	7.75×10^{-8}	3.32×10^{-6}	5.13×10^{-5}
MPSO	2.32×10^{-9}	8.68×10^{-9}	6.98×10^{-8}

Table 4. Comparison of GA, BPSO and MPSO in offline identification for N = 2500

	SSE		
	Best	Mean	Worst
GA	7.24×10^{-6}	2.31×10^{-3}	1.12×10^{-2}
BPSO	8.12×10^{-7}	5.21×10^{-6}	4.65×10^{-5}
MPSO	5.43×10^{-9}	1.95×10^{-8}	7.63×10^{-8}

Table 5. Iterations and time required

Algorithm	GA	BPSO	MPSO
Iterations	182	121	49
Elapse Time (*sec*)	28.21	10.34	4.69

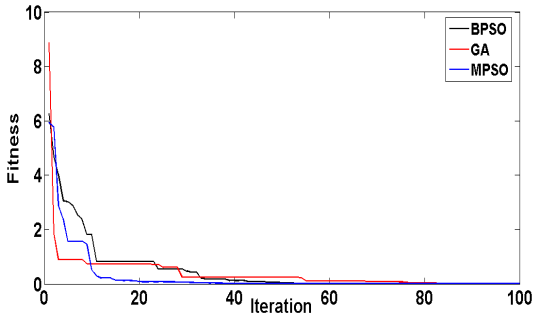
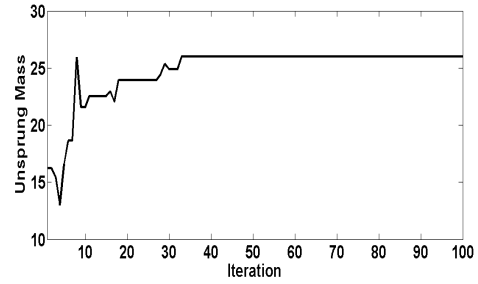
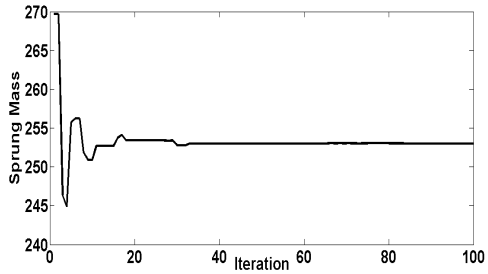


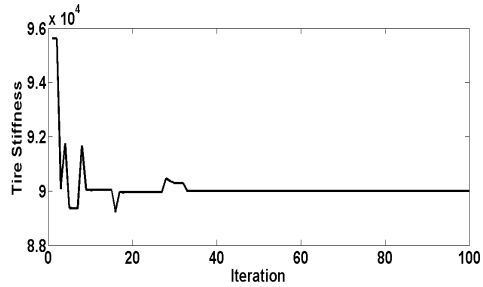
Figure 3. Comparison of convergence speed for GA, BPSO and MPSO



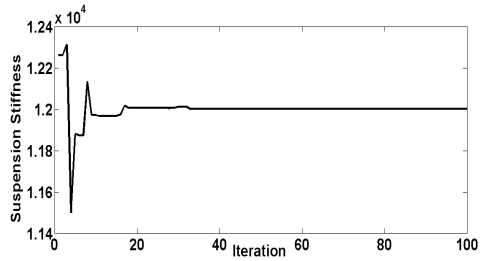
(a) Unsprung mass m_1



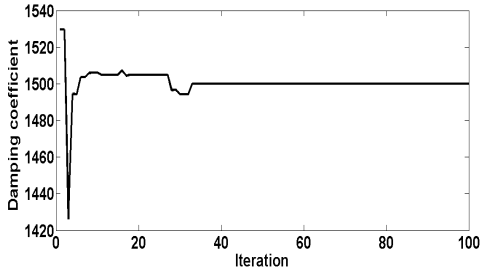
(b) Sprung mass m_2



(c) Tire stiffness k_1



(d) Suspension stiffness k_2



(e) Damping coefficient b

Figure 4. MPSO process for identification of suspension parameters

aforementioned algorithms in terms of accuracy and convergence speed.

5.2 Online Parameter Identification

Based on the pervious section, the MPSO has more accuracy and faster convergence speed than GA and BPSO in off-line identification. Because of this, the proposed method is applied for online identification of suspension system parameters. During online simulation, the sampling frequency is set to 100 kHz such that 1000 pairs of data are sampled within 0.01 msec in each period to form a data set. If a change in the model parameter is detected by sentry particle in a period, the MPSO continues to run. When the fitness of global best becomes lower than a threshold, the simulation for this period is then stopped. There will be no MPSO iteration unless another change in system parameter detect.

Figure 5 shows the fitness evaluation of the proposed method when some changes in system parameters are occurred. First nominal values of parameters are used and MPSO detects these parameters after 37 iterations for a threshold 10^{-5}. If changes in parameters occur the MPSO algorithm runs further. To show the performance of the proposed method in tracking time-varying parameters, two sudden changes are applied to suspension parameters. At the first stage, damping coefficient is changed from 1500 to 1550. At the second stage, tire stiffness is varied from 90000 to 95000. It can be seen that after the first change the algorithm detects new optimal parameters after only 19 iterations. And, after the second change the algorithm finds the new optimal parameters after only 27 iterations. It can see that the proposed method can track any change in parameters. Also since the particles are scattered around the previous global optimum depending on the values of changes in parameters, the new global optimum is found fast. **Figures 6** and **7** show the online identification results of the proposed algorithm when k_1 and b are considered as time varying parameters. It can be seen that the proposed approach can identify time-varying parameters successfully. The dashed lines in **Figures 5–7** signify the moment that the sentry particle has detected some change in system parameters.

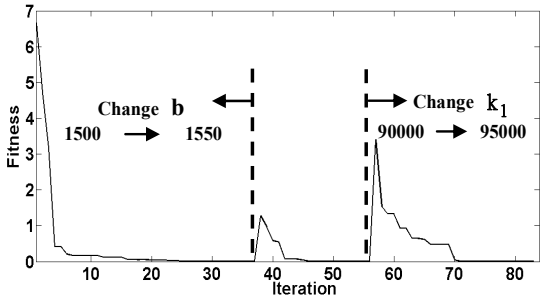


Figure 5. MPSO process in online identification of suspension system

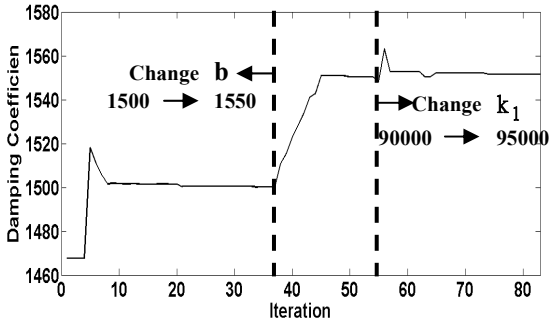


Figure 6. Identifying a time-varying damping coefficient parameter by MPSO

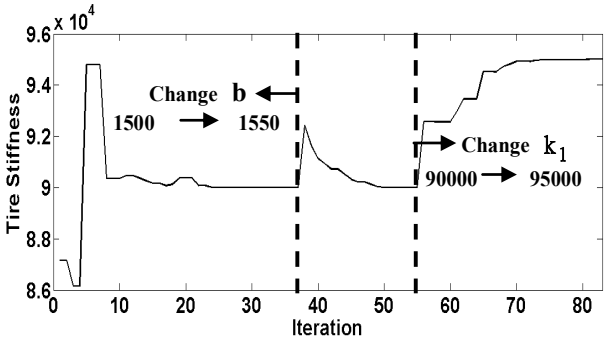


Figure 7. Identifying a time-varying tire stiffness parameter by MPSO

6. Conclusions

A quarter-car model of suspension system was used to show the effectiveness of MPSO in system identification. It has been shown that MPSO is superior to GA and BPSO in offline identification. Owing these attractive features, MPSO is applied to online identification. The estimated parameter will be updated only if a change in system parameters is detected. Thus, the proposed algorithm is a promising particle swarm optimization algorithm for system identification. Future works in this area willinclude considering variable parameters in nonlinear suspension model.

7. Acknowledgments

Our Sincere thanks to Mr. Hamidreza Modarress for his helpful comments and his advice to improve this research work.

REFERENCES

[1] L. Ljung, "System identification: Theory for the user," Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[2] K. Godfrey and P. Jones, "Signal processing for control," Springer-Verlag, Berlin, 1986.

[3] S. A. Billings and H. Jamaluddin, "A comparison of the back propagation and recursive prediction error algorithms for training neural networks," Mechanical Systems and Signal Processing, Vol. 5, pp. 233–255, 1991.

[4] K. C. Sharman and G. D. McClurkin, "Genetic algorithms for maximum likelihood parameter estimation," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Glasgow, pp. 2716–2719, 23–26 May 1989.

[5] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," International Journal of Control, Vol. 50, pp. 1873–1896, 1989.

[6] R. K. Ursem and P. Vadstrup, "Parameter identification of induction motors using stochastic optimization algorithms," Applied Soft Computing, Vol. 4, pp. 49–64, 2004.

[7] L. Liu, W. Liu, and D. A. Cartes, "Particle swarm optimization-based parameter identification applied to permanent magnet synchronous motors," Engineering Applications of Artificial Intelligence, Vol. 21, pp. 1092–1100, 2008.

[8] Z. Wang and H. Gu, "Parameter identification of bilinear system Based on genetic algorithm," Proceedings of the International Conference on Life System Modeling and Simulation, Shanghai, pp. 83–91, 14–17 September 2007.

[9] M. Ye, "Parameter identification of dynamical systems based on improved particle swarm optimization," Intelligent Control and Automation, Vol. 344, pp. 351–360, 2006.

[10] M. Dotoli, G. Maione, D. Naso, and B. Turchiano, "Genetic identification of dynamical systems with static nonlinearities," Proceedings of the IEEE Mountain Workshop on Soft Computing in Industrial Applications, Blacksburg, pp. 65–70, 2001.

[11] M. M. Fateh and S. S. Alavi, "Impedance control of an active suspension system," Mechatronics, Vol. 19, pp. 134–140, 2009.

[12] H. Du and N. Zhang, "H_∞ control of active vehicle suspensions with actuator time delay," Journal of Sound and Vibration, Vol. 301, pp. 236–252, 2007.

[13] S. J. Huang and H. Y. Chen, "Adaptive sliding controller with self-tuning fuzzy compensation for vehicle suspension control," Mechatronics, Vol. 16, pp. 607–622, 2006.

[14] C. Lauwerys, J. Swevers, and P. Sas, "Robust linear control of an active suspension on a quarter car test-rig," Control Engineering Practice, Vol. 13, pp. 577–586, 2005.

[15] H. Peng, R. Strathearn, and A. G. Ulsoy, "A novel active suspension design technique e-simulation and experimental results," Proceedings of the American Control Conference, Albuquerque, pp. 709–713, 4–6 June 1997.

[16] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," Proceedings of the IEEE International Conference on Neural Networks, Perth Vol. 4, pp. 1942–1948, 1995.

[17] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," Proceedings of the Conference on Evolutionary Computation, Alaska, pp. 69–73, 4–9 May 1998.

[18] A. Ratnaweera and S. K. Halgamuge, "Self-organizing

hierarchical particle swarm optimizer with time-varying acceleration coefficient," IEEE Transactions on Evolutionary Computation, Vol. 8, pp. 240–255, 2004.

[19] X. Yang, J. Yuan, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation," Applied Mathematics and Computation, Vol. 189, pp. 1205–1213, 2007.

[20] N. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," Proceedings of 2003 IEEE Swarm Intellihence Symposium, Indianapolis, pp. 72–79, 24–26 April 2003.

[21] M. Lovbjerg, T. K. Rasmussen, and T. Krink, "Hybrid particle swarm optimizer with breeding and subpopulations," Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, pp. 126–131, 7–11 July 2001.

[22] J. Riget and J. S. Vesterstroem, "A diversity-guided particle swarm optimizer-the ARPSO," Technical Report 2002–02, EVA Life, Department of Computer Science, University of Aarhus, pp. 1–13, 2002.

