# Particle Swarm Optimization for Adaptive IIR Filter Structures

D. J. Krusienski and W. K. Jenkins
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA

*Abstract* - **This paper introduces the application of particle swarm optimization techniques to infinite impulse response (IIR) adaptive filter structures. Particle swarm optimization (PSO) is similar to the genetic algorithm (GA) in that it performs a structured randomized search of an unknown parameter space by manipulating a population of parameter estimates to converge on a suitable solution. Unlike the genetic algorithm, particle swarm optimization has not emerged in adaptive filtering literature. Both techniques are independent of the adaptive filter structure and are capable of converging on the global solution for multimodal optimization problems, which makes them especially useful for optimizing IIR and nonlinear adaptive filters. This paper outlines PSO and provides a comparison to the GA for IIR filter structures.**

## I. Introduction

Many real-world systems that employ adaptive signal processing, such as channel equalization, speech synthesis and recognition, acoustical modeling, etc. are recursive in nature and would greatly benefit from implementing adaptive IIR processing. The primary reason that adaptive IIR filtering is seldom used in practice is the lack of practical, efficient, and robust global optimization algorithms. This paper introduces a novel algorithm named particle swarm optimization (PSO) for adaptive IIR filtering. Population based algorithms, such as PSO and the GA, are envisioned to receive increasing attention as parallel computing technology continues to progress to the forefront of information processing.

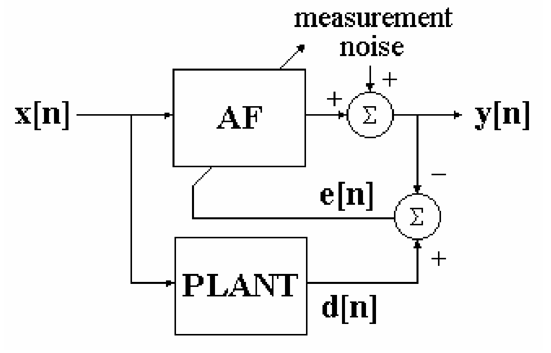

Fig. 1. Adaptive System Identification Configuration

For adaptive filtering problems such as system identification shown in Figure 1, the adaptive filter (AF) attempts to iteratively determine an optimal model for the unknown system (PLANT) based on some function of the error between the output of the AF and the output of the plant. The optimal model or solution is attained when this function of the error is minimized. In cases where the unknown plant contains feedback, a simple finite impulse response (FIR) adaptive filter of reasonable length may not be sufficient to provide an adequate model of the system. In these cases, it is only natural to model the unknown system using an IIR adaptive filter.

The drawback to IIR adaptive filter structures is that they produce error surfaces that inherently tend to be multimodal. When the error surface is multimodal, local optimization techniques that work well for FIR adaptive filters, such as versions of gradient descent algorithms, are not suitable because they are likely to get trapped in a local minimum solution. A few modifications to gradient decent algorithms exist that can improve the performance, such as adding noise to the gradient calculation [11] to make it more likely to escape from a local minima, or using the equation error [11] adaptation to transform the error surface to be unimodal. However, adding noise to the gradient can slow convergence and does not guarantee escape from local minima, and the equation error formulation can lead to a biased solution.

An alternative to gradient descent based techniques is a structured stochastic search of the error space. These types of global searches are structure independent because a gradient is not calculated and the AF structure does not directly influence the parameter updates – aside from the error computation. Due to this property, these types of algorithms are potentially capable of globally optimizing any class [7] of adaptive filter structures or objective functions. Several structured stochastic search approaches have appeared in the IIR adaptive filtering literature, most notably simulated annealing [9] and evolutionary algorithms such as the GA [8][9][10][12][13][14]. PSO is another structured stochastic search algorithm that has recently gained popularity for optimization problems. This paper outlines PSO, applies it with modifications to IIR adaptive filtering, and provides examples for comparison to the GA.

Although this paper concentrates on the application of PSO methods to IIR adaptive filter structures, the same algorithms are also effective for nonlinear AF structures and neural networks [3] that are also prone to multimodal error surfaces. The use of PSO methods for nonlinear structures will be the subject of another paper.

## II. Particle Swarm Optimization

Particle swarm optimization was first developed in 1995 by Eberhart and Kennedy [6], rooted on the notion of swarm intelligence of insects, birds, etc. Similar to the GA, PSO begins with a random population of individuals, referred to here as a swarm of particles. As with the GA, each particle in the swarm is a different possible set of the unknown parameters to be optimized. Each particle represents a point in the solution space that has a relative fitness determined by evaluating the parameters with respect to a predetermined fitness function that has an extremum at the desired optimal solution. The goal is to efficiently search the solution space by swarming the particles toward the best fit solution encountered in previous iterations with the intent of encountering better solutions through the course of the process and eventually converging on a single minimum error solution.

The standard PSO algorithm begins by initializing a random swarm of M particles (an adequate M is dependent on the dimensionality of the problem), each having R unknown parameters to be optimized. At each iteration, the fitness of each particle is evaluated according to the selected fitness function. The algorithm stores and progressively replaces the most fit parameters of each particle (*pbest*$_i$, *i=1,2,...,M*) as well as a single most fit particle (*gbest*) as better fit parameters are encountered. The parameters of each particle ($p_i$) in the swarm are updated at each iteration ($n$) according to the following equations:

$$\begin{aligned}\overline{vel}_i(n) &= w * \overline{vel}_i(n-1) \\ &\quad + acc_1 * diag\left[e_1, e_2, ..., e_R\right]_{i1} * (gbest - p_i(n-1)) \\ &\quad + acc_2 * diag\left[e_1, e_2, ..., e_R\right]_{i2} * (pbest_i - p_i(n-1))\end{aligned} \tag{1}$$

$$p_i(n) = p_i(n-1) + \overline{vel}_i(n) \tag{2}$$

where

$\overline{vel}_i(n)$=velocity vector of particle i
$e_r$=random values $\in$ (0,1)
$acc_1$=acceleration coefficient toward *gbest*
$acc_2$=acceleration coefficient toward *pbest*$_i$
$w$=inertia weight

The trajectory of each particle is influenced in a direction determined by the previous velocity and the location of *gbest* and *pbest*$_i$. The acceleration constants are typically chosen in the range 0-2 and control the relative influence toward *gbest* and *pbest*$_i$ respectively by scaling each resulting distance vector. The two acceleration coefficients combined form what is analogous to the step size of an adaptive algorithm. Acceleration coefficients closer to 0 will produce fine searches of a region, while coefficients closer to 1 will give a lesser search and faster convergence. Setting the acceleration greater than 1 allows the particle to possibly over-step *gbest* or *pbest*, resulting in a broader search. The random $e_i$ vectors have R different components, which are randomly chosen in the range 0-1. This allows the particle to take constrained randomly directed steps in a bounded region between *gbest* and *pbest*$_i$. The acceleration coefficients should be chosen in conjunction with the random $e_i$ components for a desired average step size.

The inertia weight controls the influence of the previous velocity. It is typically set to decay from 1 to 0 during some adequate interval in order to allow the algorithm to converge on *gbest*. A single particle update is graphically illustrated in two dimensions in Figure 2. The new particle coordinates can lie anywhere within the bounded region, depending upon the weights and random components associated with each vector.

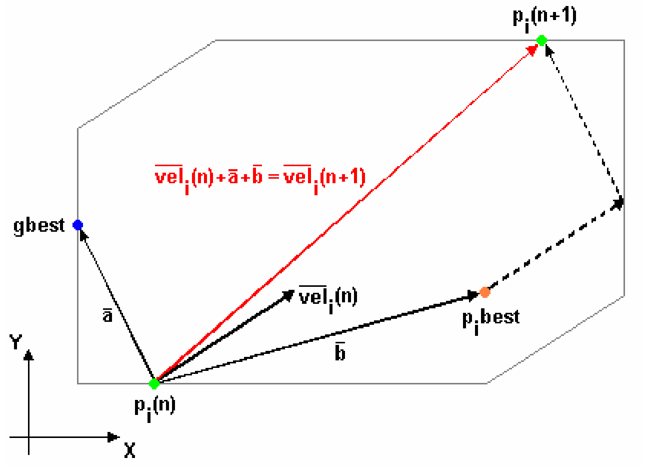

Fig. 2. Example of the possible search region for a single particle

As new *gbests* are encountered during the update process, all other particles begin to swarm toward the new *gbest*, continuing to search along the way. The search regions continue to constrict as new *pbest*$_i$*s* are encountered. The algorithm is terminated when all of the particles in the swarm have converged to *gbest* or a suitable minimum error condition is met.

### *2.1 PSO for Adaptive IIR Filtering*

For batch processing type optimization problems, the particle fitness would be evaluated at each iteration using the entire input data. In typical on-line adaptive filtering problems, the entire input data are not available or the data stream is too lengthy to process in an efficient manner. Therefore, the input data must be processed and evaluated in blocks, producing an estimate of the actual error. This estimate can be improved by averaging the error estimates over a window of previous input data.

In adaptive filtering, the mean squared error (MSE) between the output of the unknown system and the output of the AF is the typical cost function that is used for the fitness evaluation of each particle in the on-line form of PSO. For the IIR adaptive system identification configuration as shown in Figure 1, the windowed MSE cost function is as follows:

$$J_i(n) = \frac{1}{N}\left[\sum_{k=0}^{N}(d(n-k) - y_{k,i}(n))^2\right] \quad (3)$$

$$y_{k,i}(n) = \sum_{m=1}^{Q-1} p_i^m(n)y(n-k-m) + \sum_{m=0}^{P-1} p_i^{m+Q}(n)x(n-k-m) \quad (4)$$

where $Q$ and $P$ are the number of numerator and denominator terms respectively, $N$ is the length of the window over which the error is averaged, and the $p_i$'s are the tap weights, indexed by the superscript. When *J(n)* is minimized, the AF parameters provide the best possible representation of the unknown system. Because cost function is based on a windowed recursion for IIR filters, the best estimates are obtained by initialing the recursions with the desired signal.

### *2.2 Modifications for Adaptive IIR Filtering*

Some suggested modifications and variations on the standard PSO algorithm to improve the overall efficiency were outlined in a previous paper [7]. A few of these modifications, in addition to a new modification are summarized below for the IIR case.

1) A recently discovered modification, not discussed in [7], that improves the efficiency and speed of the search is to independently adjust the inertia weight of each particle according to if a new *pbest* is encountered. The concept is that the inertia weight should be increased when a new *pbest* is encountered to keep the particle in a likely decent. For the opposite reason, if the particle does not encounter a new *pbest*, its inertial influence should be less. This modification, however, does not prevent the hill-climbing capabilities of PSO, it merely increases the influence of potentially fruitful inertial directions, while decreasing the influence of potentially unfavorable inertial directions.

2) To prevent an outlying *gbest* from being approached from limited directions, when a new *gbest* is encountered, randomly selected particles can be re-randomized about the new *gbest*. This will act to ensure that the region around *gbest* is searched from all directions, while keeping a portion of the swarm searching globally.

3) In the early stages of the algorithm, particles closest to *gbest* tend to converge quickly and become stagnant. This can be eliminated by slightly varying the random parameters of each particle at every iteration, similar to mutation in evolutionary algorithms. This will have little effect on particles distant from *gbest* because this random influence should be relatively small compared to the random update of equation (1). However, this will eliminate any stagnant particles, creating a finer search about *gbest*.

4) The issue of premature convergence on a local minimum is occasionally inevitable, depending on the characteristics of the error surface or other constraints. The likelihood of this can be decreased by continually re-randomizing a randomly selected portion of the particles over the entire parameter space and allowing them to converge. This will in effect continually generate unique search paths, which can increase the probability of finding the global optimum. This continuous probing of the space is also beneficial when tracking a non-stationary input or dynamic plant [1][5].

5) Population based searches are inherently less likely to produce unstable IIR filters due to the number of estimates available at any given iteration. To further prevent the possibility of instability in the adaptive IIR filter, the system can be converted to an equivalent parallel second order structure and the search can be restricted to the stable region of the parameter space using the stability triangle [11].

In order to ensure convergence of the swarm, the variance of the mutation and selected re-randomization distributions must decrease according to some schedule that will allow a sufficient search of the space [7]. The re-randomization and acceleration schedules can be coordinated to optimize the convergence speed and search efficiency.

## III. Simulation Examples

In the following examples, the properties of PSO and a modified version of PSO (MPSO) are compared to a real-encoded GA for two IIR system identification problems. All algorithms were initialized with the same population of real-valued parameters and allowed to evolve. The window length, $N$, was set to 100 in each case. For each simulation, the MSE is averaged over 50 successful trials in which the algorithm converged to the neighborhood of the global optimum. The specifics of each algorithm are as follows:

**PSO:** The standard PSO algorithm is implemented with both acceleration constants weighted equally at 1, giving an average step size of approximately 0.5. The acceleration constant was chosen to give a reasonable balance between the search quality and convergence speed for each case.

**MPSO:** The modified PSO algorithm uses the standard PSO algorithm as a base, implementing the first 4 modifications suggested earlier. As with standard PSO, both acceleration constants are weighted equally at 1.

**GA:** The genetic algorithm uses a ranked elitist strategy, where the 6 fittest members of the population are used to generate offspring, which replace the remaining least fit members of the population. This scheme was suggested in [8] to enhance the rate of convergence. For each offspring, two of the 6 parents are selected randomly and the crossover is performed by a random weighted average of each parent's coefficients. A mutation rate of 0.125 is used, where mutations are randomly selected uniformly on the range of 0-0.5 initially, with a decreasing variance to aid the convergence of the population.

### *3.1 Matched order, White noise input, SNR=40dB*

For this example the plant, given below, is a second order pole-zero filter taken from [14].

$$PLANT = \frac{1.25z^{-1} - 0.25z^{-2}}{1 - 0.3z^{-1} + 0.4z^{-2}} \qquad AF = \frac{p_i^1 + p_i^2 z^{-1} + p_i^3 z^{-2}}{1 + p_i^4 z^{-1} + p_i^5 z^{-2}}$$

The experimental results for this example using a population of 25 and 50 are shown in Figures 3 and 4 respectively.

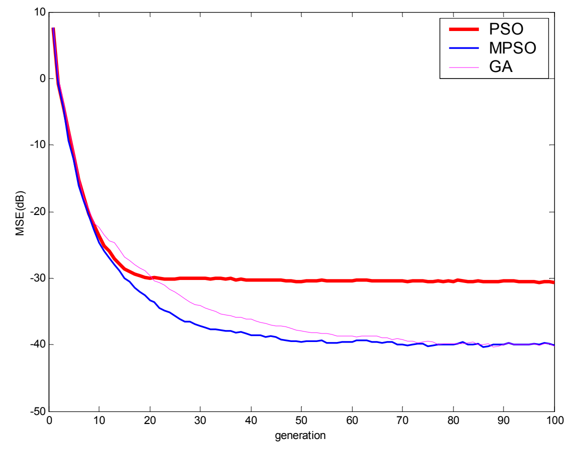

Fig. 3. Example 3.1, Population=25

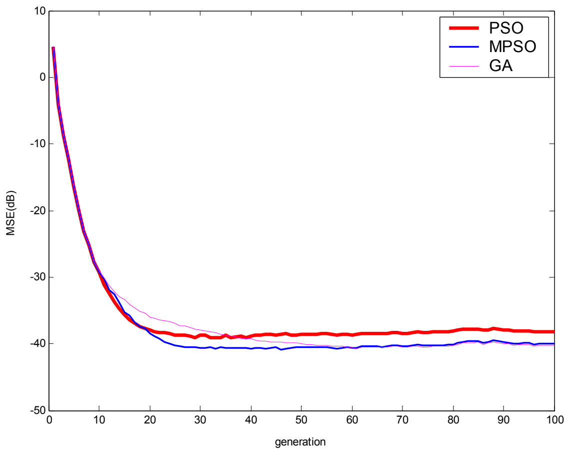

Fig. 4. Example 3.1, Population=50

### *3.2 Matched order, Colored noise input, SNR=20dB*

For this example the plant, given below, is an example taken from [8] and [13].

$$PLANT = \frac{1}{1 - 1.4z^{-1} + 0.49z^{-2}} \qquad AF = \frac{p_i^1}{1 + p_i^2 z^{-1} + p_i^3 z^{-2}}$$

The adaptive filters use a colored input generated by filtering white noise by the FIR filter $H_c(z)=(1-0.7z^{-1})^2(1+0.7z^{-1})^2$, creating a bimodal error surface. The experimental results for this example using a population of 25 and 50 are shown in Figures 5 and 6 respectively.

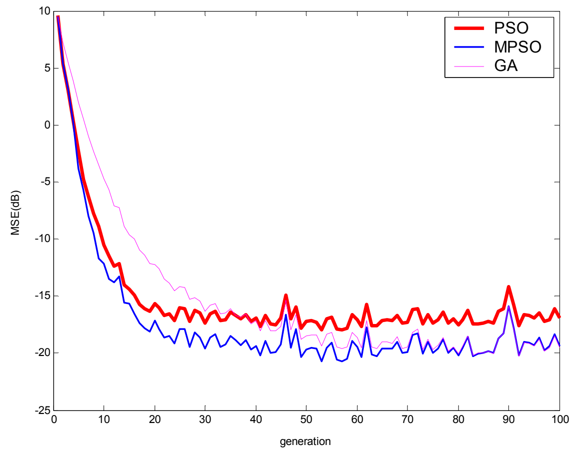

Fig. 5. Example 3.2, Population=25

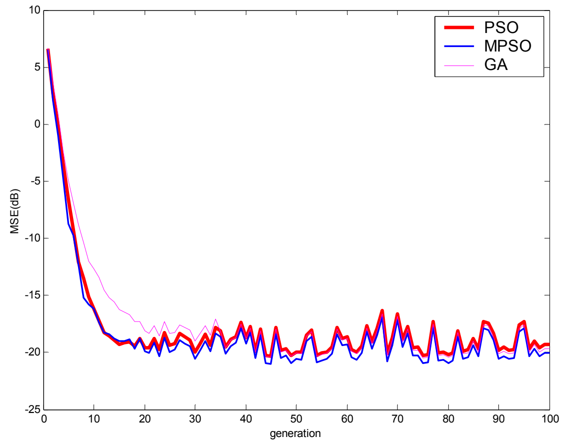

Fig. 6. Example 3.2, Population=50

### *3.3 Reduced order, Colored noise input*

For this example the plant, given below, is an example taken from [8] and [13].

$$PLANT = \frac{1}{(1 - 0.6z^{-1})^3} \qquad AF = \frac{p_i^1}{1 + p_i^2 z^{-1} + p_i^3 z^{-2}}$$

The adaptive filters use a colored input generated by filtering white noise by the FIR filter $H_c(z)=(1-0.6z^{-1})^2(1+0.6z^{-1})^2$. This, in combination with the reduced order, creates a bimodal error surface. The experimental results for this example using a population of 25 and 50 are shown in Figures 7 and 8 respectively.

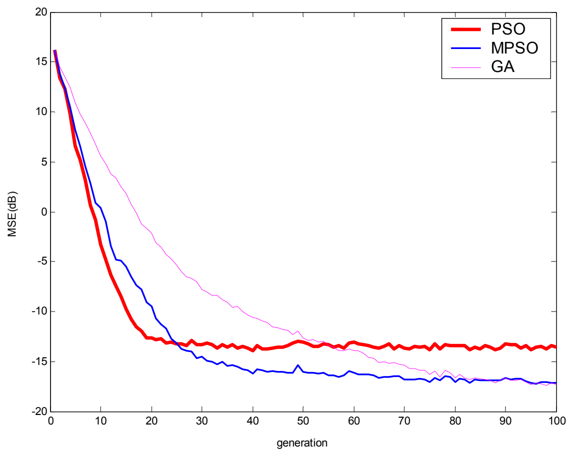

Fig. 7. Example 3.3, Population=25

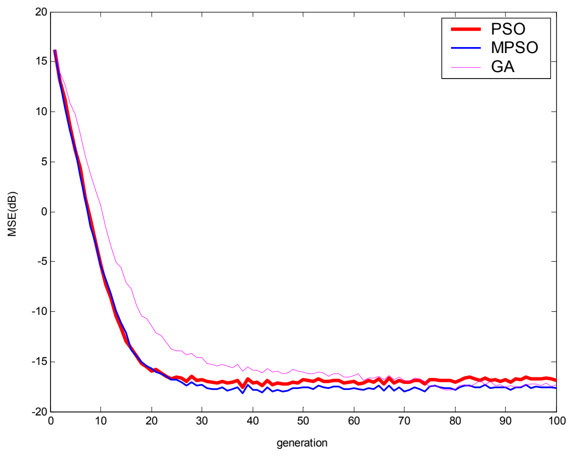

Fig. 8. Example 3.3, Population=50

*3.4 Matched high-order, White noise input, SNR=40dB*

For this example the plant, given below, is a fifth-order low-pass Butterworth filter example taken from [13].

$$PLANT = \frac{0.1084+0.5419z^{-1}+1.0837z^{-2}+1.0837z^{-3}+0.5419z^{-4}+0.1084z^{-5}}{1+0.9853z^{-1}+0.9738z^{-2}+0.3864z^{-3}+0.1112z^{-4}+0.0113z^{-5}}$$

$$AF = \frac{p_i^1+p_i^2z^{-1}+p_i^3z^{-2}+p_i^4z^{-3}+p_i^5z^{-4}+p_i^6z^{-5}}{1+p_i^7z^{-1}+p_i^8z^{-2}+p_i^9z^{-3}+p_i^{10}z^{-4}+p_i^{11}z^{-5}}$$

The experimental results for this example using a population of 50 and 100 are shown in Figures 9 and 10 respectively.

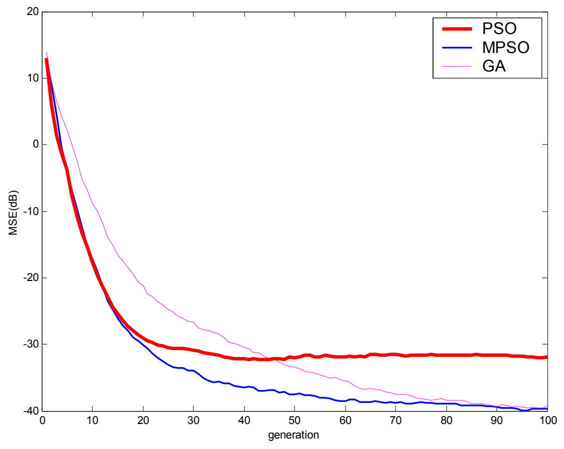

Fig. 9. Example 3.4, Population=50

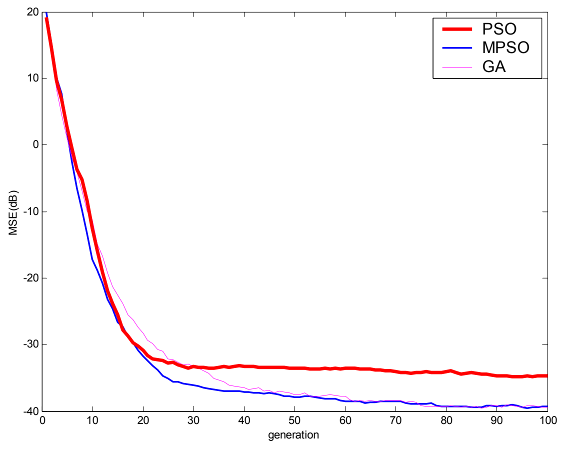

Fig. 10. Example 3.4, Population=100

## IV. Discussion

For on-line adaptive filtering applications, the primary performance considerations for an adaptive algorithm are the rate of convergence and the minimum mean squared error achieved. From the simulations, it is observed that the particle swarm algorithms are able to converge very rapidly when the error surface is relatively compliant. This is the fundamental advantage of particle swarm algorithms for on-line adaptive filtering. For comparison purposes, the particular variation of the genetic algorithm was chosen and tuned to produce a rate of convergence comparable to that of particle swarm. Since the GA doesn't have an explicit step size, the convergence rate can only be controlled to a limited extent through the crossover and mutation operations, and the algorithm must evolve at its own intrinsic rate.

Though standard PSO exhibits a fast convergence initially, it fails to improve further because the swarm quickly becomes stagnant, converging to a suboptimal solution. However, with the same set of algorithm parameters, the MPSO particles do not stagnate, allowing it to reach the noise floor along with the GA. Smaller acceleration coefficients can be used with standard PSO to enable it to reach the noise floor, forsaking the rapid convergence rate. By introducing a simple mutation type operator, MPSO can retain the optimal convergence rate, while still achieving the noise floor.

The examples having the larger population illustrate the case where the population size is sufficient and all of the algorithms rarely become trapped in a local minimum. In cases where the error surface becomes increasingly complex or the population size is deficient, the likelihood of becoming trapped in a local minimum is increased. This is evident in the examples where the population size is reduced by a factor of two. With the same exact algorithms, the rate of convergence begins to decrease due to the reduced search capacity, while the number of instances the algorithms converge on a local minimum increases. Though the plots are generated using only successful trials, for the decreased population sizes MPSO encountered local minima approximately 2% of the trials, the GA approximately 6%, and PSO approximately 20% in the examples with multimodal error surfaces. Again, at the expense of convergence speed, the acceleration coefficients of standard PSO could be adjusted to give a broader search, lessening the probability of converging on a local minimum. By intelligently re-randomizing a small portion of the particles insignificant to the global search, MPSO remains robust because it is able to retain its convergence speed while avoiding local minima using a fewer number of particles. This can offer considerable savings in cases where computational complexity is an issue.

## REFERENCES

[1] Carlisle, A. and Dozier, G. "Adapting Particle Swarm Optimization to Dynamic Environments," Proceedings of the International Conference on Artificial Intelligence. Las Vegas, Nevada, 2000.

[2] Eberhart, R. C. and Shi, Y. "Comparison between genetic algorithms and particle swarm optimization," Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, San Diego, CA. 1998.

[3] Eberhart, R. C. and Shi, Y. "Evolving artificial neural networks," Proceedings of International Conference on Neural Networks and Brain, 1998, Beijing, P. R. China. pp. PL5-PL13, 1998.

[4] Haykin, S., *Adaptive Filter Theory,* 4th ed. Prentice Hall, 2001.

[5] Hu, X. and Eberhart, R. C. "Adaptive particle swarm optimization: detection and response to dynamic systems," Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002.

[6] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm intelligence* San Francisco: Morgan Kaufmann Publishers, 2001.

[7] Krusienski, D.J. and Jenkins, W.K., "Adaptive Filtering Via Particle Swarm Optimization," ," *Proc. 37th Asilomar Conf. on Signals, Systems, and Computers*, November 2003.

[8] Mars, P., Chen, J.R., Nambiar, R., *Learning Algorithms: Theory and Applications in Signal Processing, Control, and Communications*, CRC Press, Inc., 1996.

[9] Nambiar, R. and Mars, P., "Genetic and Annealing Approaches to Adaptive Digital Filtering," *Proc. 26th Asilomar Conf. on Signals, Systems, and Computers*, vol. 2, pp. 871-875, Oct. 1992.

[10] Ng, S.C., Leung, S.H., Chung, C.Y., Luk, A., and Lau, W.H., "The Genetic Search Approach," *IEEE Signal Processing Magazine*, pp. 28-46, November 1996.

[11] Shynk, J.J., "Adaptive IIR Filtering," *IEEE ASSP Magazine*, pp. 4-21, April 1989.

[12] Tang, K.S., Man, K.F., He, Q. "Genetic Algorithms and their Applications," *IEEE Signal Processing Magazine*, pp. 22-37, November 1996.

[13] White, M.S. and Flockton, S.J., Chapter in *Evolutionary Algorithms in Engineering Applications*, Editors: D. Dasgupta and Z. Michalewicz, Springer Verlag, 1997.

[14] Yao, L., Sethares, W.A., "Nonlinear Parameter Estimation via the Genetic Algorithm," *IEEE Transactions on Signal Processing*, vol.42, April 1994.