Arabian Journal for Science and Engineering (2019) 44:2177–2184
https://doi.org/10.1007/s13369-018-3387-8

RESEARCH ARTICLE - ELECTRICAL ENGINEERING

A Hybrid Particle Swarm Optimization Technique for Adaptive Equalization

Ali A. Al-Shaikhi[1] · Adil H. Khan[2] · Ali T. Al-Awami[1] · Azzedine Zerguine[1]

Received: 26 December 2017 / Accepted: 3 June 2018 / Published online: 12 June 2018

Abstract
Adaptive equalization mitigates the distortions caused by radio channels. The least mean square (LMS) and the recursive least squares (RLS) algorithms are used for such purpose. Recently, particle swarm optimization (PSO) algorithms such as PSO using a linear time decreasing inertia weight (PSO-W) and the PSO using constant constriction factor (PSO-CCF) were shown to be very effective in handling systems having nonlinear behavior. However, these algorithms can be trapped in local minima. This paper presents a new PSO-based algorithm called the hybrid PSO (HPSO) that is capable to handle such problems. The HPSO includes the randomization of particles to improve the search capacity of the swarm, which in turn reduces the probability of being trapped in some local minima. It also adapts the inertia weight assignment to the particles. Extensive simulation results are conducted to confirm the consistency in the performance of the HPSO algorithm in different scenarios. The proposed HPSO secures the minimum steady-state error as compared to LMS and other PSO-based algorithms in both nonlinear and linear channels. Finally, the proposed HPSO algorithm shows a great improvements in Bit Error Rate and convergence rate.

Keywords Particle swarm optimization (PSO) · Adaptive channel equalization · Hybrid PSO · LMS Algorithm · Evolutionary algorithms (EAs)

1 Introduction

Adaptive equalization can be used to improve digital data transmission over unknown channels; it is used mainly to mitigate the effect of intersymbol interference (ISI) [1]. The main use of adaptive equalization is in systems with high speed of communications and especially when such systems do not use frequency division multiplexing or differential modulation schemes. An equalizer is the most important and expensive part of the demodulator for any communication system as it usually consumes more than 80% of the whole computation required to demodulate any signal [2]. There are many algorithms currently used for adaptive channel equalization, to name a few, the least mean square (LMS), the recursive least squares (RLS), their variants [3], and recently algorithms based on the particle swarm optimization (PSO) technique [4–7]. The LMS is considered as the most commonly used algorithm for channel equalization, this is due mainly to its low computational cost. However, particle swarm optimization is nowadays applied in many optimization research problems. This heuristic approach of finding the optimum value for any cost function lies under the category of swarm intelligence. Kennedy and Eberhart first proposed this algorithm [8], in which it is applied to a simplified social model. They took the mental state of every individual as the position of the particle, where each individual will possess its own mental state and attitude [9].

The PSO technique is better than other evolutionary algorithms (EAs) in the sense that the system is initialized by random solutions for the optima by updating the generations without mutation or crossover. The potential solutions here,

✉ Ali A. Al-Shaikhi
shaikhi@kfupm.edu.sa

Adil H. Khan
akhan@pmu.edu.sa

Ali T. Al-Awami
aliawami@kfupm.edu.sa

Azzedine Zerguine
azzedine@kfupm.edu.sa

[1] Department of Electrical Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

[2] Department of Electrical Engineering, Prince Mohammad bin Fahd University, Al Khobar 31952, Saudi Arabia

which are the particles, fly through the problem by following the current optimum particle. An important advantage of PSO over EAs is that it has memory, meaning that each and every particle will remember the best position, or best solution, it has achieved so far and also it will remember the best position (solution) of the group, known as the global best. PSO is also more suitable to handle time-varying problems. Another advantage of PSO over EAs is that the number of members of the population will remain fixed through the whole process. Therefore, it can be stated that PSO uses productive collaboration among particles, which is not the case in other different artificial algorithms where the main idea is that only the fittest members will survive [9].

Different variations of the PSO technique were introduced in order to improve the performance of different applications/scenarios. For example, many researches discussed the effect of inertia weights on the performance of PSO, as the velocity is multiplied by this inertia weight factor before the velocity updating process begins; therefore, the inertia weight actually controls the velocity of the particles [10]. Another version was introduced in [11], where a constriction factor was used instead of the inertia weight factor, and this constriction factor is used to ensure that the particles should be limited to their present velocity before updating. The advantage of this algorithm is the enhanced convergence rate, but for multi-modal problems; however, there is a chance that due to its swift convergence rate, it might get trapped in some local minima.

Many researches have discussed this critical issue, while showing all the analysis regarding the tuning of the PSO parameters. Among them are the ones proposed by Shi and Eberhart [10,12,13], Carlisle and Dozier [14], Angeline [15], and El Gallad [16]. Another development was proposed to reduce the training time for the PSO. This is known as cooperative particle swarm optimizer (CPSO) [17,18]. In this technique, the whole swarm will be divided into two different small-size swarms where each one will be driven by a separate PSO. The effect of the size chosen of the two swarms is discussed in [19]. Parsopoulos and Vrahatis [20] used the PSO algorithm for the first time to solve problems of multi-objective function. Another version of the PSO algorithm, proposed by Xu [21], named the extended PSO (EPSO). In this method, at each iteration while updating the velocity of each particle, the algorithm used both global and local best positions, whereas previously only global positions were used for velocity updating process. This methodology combines simultaneously the best effects of both local and global best positions. A very useful work, which enlists the developments and all the applications and resources of PSO algorithm, is presented in [22]. It also presents all the arrays of functions and different types of it where PSO can be a better choice. Another very important modification, proposed in [23], is made by considering the inertia weight as a function of improvement where improvement is defined here as any achievement produced by any particle in the present iteration compared to its previous one. There are some other techniques proposed to overcome some critical issues related to the PSO algorithm, such as enhancing the convergence speed by using Gaussian distribution instead of uniform distribution when selecting the random directions of particles or vectors.

Every one of the above-listed algorithms addressed a deficiency to improve their performance. In this work, we propose a new hybrid PSO algorithm, the HPSO algorithm which includes the randomization of particles to improve the search capacity of the swarm. This will introduce more socialized behavior among particles to reduce the probability of being trapped in some local minima. Furthermore, it adapts the inertia weight assignment to the particles. Extensive simulation results are conducted to confirm the consistency in the performance of the HPSO algorithm in different scenarios.

The paper is organized as follows. Following the Introduction, Sect. 2 highlights some of the most popular particle swarm optimization algorithms, while the proposed hybrid PSO algorithm is presented in Sect. 3. The simulation results are presented in Sect. 4. Finally, some conclusions are given in Sect. 5.

2 Particle Swarm Optimization Algorithms

The PSO algorithm works on the basis of its population members (particles), which are collectively known as swarm. These particles work collectively to optimize a certain cost function, which should be real valued and have a specific number of dimensions. Each particle in the swarm will have a specific position which is known as a potential solution to the targeted optimization problem in the solution space. The PSO algorithm will approach the most optimal solution by applying specific changes to the existing set of solution, and these changes will be applied through probabilistic and iterative modifications. Every particle will have two basic parameters, position and velocity, where position represents the current solution and velocity is used to speed-up/slowdown the particle to approach the optimum value. Both of these parameters have immense importance in finding the optimum solution because large values of these parameters will cause the particles to fly away from the solution space, and small values slow the convergence rate. In the PSO algorithm, the particles are first generated randomly, and like any other optimization algorithm, they will have an objective or cost function. For channel equalization, which is considered in this work, the PSO algorithm uses the following cost or objective function:

$$J_i(l) = \frac{1}{N}\sum_{m=1}^{N}[e_{mi}(l)]^2, \tag{1}$$

where

$$e_{mi}(l) = d_{mi}(l) - y_{mi}(l), \tag{2}$$

which represents the mth error of the ith particle and it is the difference between the received signal y from the desired signal d. The symbol N is the input data window size of the equalizer. The symbol l takes the values from 1,..., L, where L is the number of iterations. This cost function will be applied to each and every particle, then the minimum among all the particles is identified and is called *global minima (best) or gbest*. Every particle memorizes the minimum value achieved so far among all its previous values, and this is known as *particle best or (pbest)*. The positions of the particles should be changed in such a way that they approach an optimized value. While generating and applying the position and velocity updates for the particles, no particle should fly out from the specified range.

The PSO algorithm starts by generating a random potential solution for each particle. Let the ith particle has D dimensions, then the solution for this particle will be represented by

$$X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iD}). \tag{3}$$

Here every particle will follow its coordinates in a multi-dimensional space, related to the most plausible solution that a specific particle has achieved so far. The fitness value, which has been achieved so far by any specific particle i, *(pbest)*, is also stored as

$$P_i = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{iD}). \tag{4}$$

As stated earlier, the PSO algorithm also keeps tracks globally. Therefore, each particle will calculate the best value globally *(gbest)* and find its location, which has been secured so far. At every iteration, l, the PSO algorithm will keep on varying the velocity of each particle, i, in the direction of its *pbest* and *gbest*, according to the following recursion [24]:

$$\begin{aligned} v_{id} = {} & w \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) \\ & + c_2 \times rand() \times \left(p_{gd} - x_{id}\right), 1 \le d \le D, \end{aligned} \tag{5}$$

where w is the inertia weight, c_1 and c_2 are acceleration constants, $rand()$, which is a uniformly distributed random number between zero and one, is used to introduce the stochastic behavior in the velocity equation, $p_{id} = pbest$, and $p_{gd} = gbest$. Ultimately, the position update equation is given by

$$x_{id} = x_{id} + v_{id}, \qquad 1 \le d \le D. \tag{6}$$

It is worth noting here that the position x is related to the velocity v by multiplying v with time t, where it is assumed that t is the unit one of time. The same principle is assumed when relating v to the acceleration. Now, the effect of each parameter, that is the inertia weight, w, and the constants c_1 and c_2, used in the PSO algorithm, on the performance and the efficiency of the algorithm, will be discussed. The inertia weight controls the momentum expression, $w \times v_{id}$, which denotes the effect of the preceding velocity of the ith particle on its present velocity. A large value of w will make the present velocity, v_{id}, large and due to this effect the particle will search a larger solution space, which will obviously help finding the global best solution, but it will slowdown the convergence rate. On the other hand, small values of w will result in a fast convergence; however, this might result in a solution that may trap the particle in a local minimum [4].

The rate, at which the particles move in the direction of their local best values, is controlled by the acceleration constants c_1 and c_2 which control the movement of the particle in the direction of the global value. When $c_1 = 0$, the particles will have global experience only, which means that the particles will not have any cognitive control but will be affected by social weight only, and hence all particles will move freely in a swarm with less probability to reach a global solution. If we assign $c_2 = 0$, then every particle will endure only self-experience, meaning it will make the decisions only by cognitive sense. The convergence rate in this case is quite high but there will be a possibility that the particles will get trapped in some local optimum value. When $c_1 = c_2 = 0$, then all particles will not be having any kind of social or cognitive experience, and this will result in a disordered movement of the particles in the swarm. Therefore, a trade-off has to be made among these parameters. In order to make sure that all particles should not fly out from the solution space, the velocities of every particle will be constrained in some selected range $[-v_{\max}, v_{\max}]$, where the value of $v_{\max}$ is a problem dependent. Here, the vector of velocity for the ith particle will be defined as

$$V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{iD}). \tag{7}$$

Random weight assignment to the acceleration will be carried out separately, and ultimately will ensure the particles to move toward *pbest* and *gbest*.

2.1 Conventionally Used PSO Algorithms for Adaptive Equalization

As can be seen from (5), the inertia weight is a constant value throughout the use of the PSO algorithm. Unlike (5), then the first change in the algorithm was made by adjusting its inertia weight as a function of time. This helps improving the speed

of convergence. In [13], a linearly decreasing time inertia weight is proposed and updated according to

$$w_l = (w_{\text{initial}} - w_{\text{final}}) \times \frac{L-l}{L-1} + w_{\text{final}}, \tag{8}$$

where w_{initial} is the initial weight, w_{final} is the final weight, L is the maximum number of iterations, and l is the current iteration. The position update equation will remain the same. This algorithm is named as the PSO-W algorithm.

As stated earlier, the inertia weight term is used to control the effect of velocity updates on both the present position and the exploration of local and global solutions in the search space. Hence, the motive for using a linearly time decreasing inertia weight parameter is that large values of inertia weight will be used at the start of the operation to make sure that the particles explore globally the search space. Near the end of the operation, smaller values are used to make sure that the particles explore only locally around a globally optimum value.

To improve more the convergence rate, instead of the inertia weight a constriction factor is used to make sure that the PSO algorithm will converge in a less number of iterations. A basic constriction factor was anticipated in [13] and is given by the following:

$$K = \frac{k}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \tag{9}$$

where $k = 2$, $\phi = c_1 + c_2$, and $\phi > 4$. This algorithm is named as the PSO-CCF algorithm. Consequently, the velocity update vector is carried out according to

$$\begin{aligned} v_{id} = K\,[&v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) \\ &+ c_2 \times rand() \times \left(p_{gd} - x_{id}\right)], \quad 1 \le d \le D. \end{aligned} \tag{10}$$

An idea of using a variable constriction factor was proposed in [4]. This variable constriction factor is shown to be governed by the following update equation:

$$k_l = k_{\min} + (k_{\max} - k_{\min}) \times \frac{L-l}{L-1}, \tag{11}$$

where L is the maximum number of iterations, l is the current iteration and $k_{\max}$ and $k_{\min}$ are the maximum and minimum values of k_l, respectively. Ultimately, the velocity update equation becomes

$$\begin{aligned} v_{id} = k_l\,[&v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) \\ &+ c_2 \times rand() \times \left(p_{gd} - x_{id}\right)], \quad 1 \le d \le D, \end{aligned} \tag{12}$$

where k_l is given by (11). Unlike (10), Equation (12) has a variable constriction factor updated every iteration in accordance with (11), which is similar to the idea of variable step size in the LMS algorithm [25]. This resulted in the PSO-VCF algorithm [4].

2.2 Identified Problems

In the previously used PSO algorithms, whenever a new *gbest* is found all particles will start to move toward it in the same general direction. Due to this behavior, there is a chance that some regions, other than this new minima discovered, will be excluded from the search space. Particles which are closer to *gbest* will tend to converge to it prematurely, i.e., there will be no update in their positions. These particles will become stagnant and will not contribute further in the search procedure. Eventually, this might result in a degraded steady-state error. In the next section, the proposed algorithm is designed to address these issues.

3 The Proposed Hybrid PSO Algorithm

Here, we propose a new hybrid PSO (HPSO) algorithm which will incorporate three techniques to address the above-stated problems, namely:

- The re-randomization of the particles around *gbest*,
- An enhanced socialized effect through parameter local best (*lbest*),
- An adaptive inertia weight assignment to the particles.

In order to enhance the diversity, every time a new *gbest* is found, re-randomize the particles around it [23]. To enhance the socialized effect, Eberhart and Kennedy in [26] proposed a version of PSO which uses local information for decision making. They made ring type topology, in which a number of particles are included, and they communicate only with each other, not with the whole swarm for velocity and position updates. In our proposed algorithm, we will incorporate this technique with global best evaluations to improve the steady-state error and this parameter will be named *lbest*. It operates just like *gbest* parameter, except it will divide the particles into a number of subgroups. Therefore, each particle will have to memorize three entities: the best position achieved by the individual particle *(pbest)*, the best position achieved so far among all particles *(gbest)*, and the best position achieved by the particle in its subgroup *(lbest)*. These two techniques will degrade the convergence rate profoundly, as in both of these techniques, particles are exploring more and more the search space. Therefore, in order to ensure that at the end of the convergence the particles should converge to the optimum value, both of these two techniques will be applied with a variance curve, which is shown in Fig. 1 and it is given by

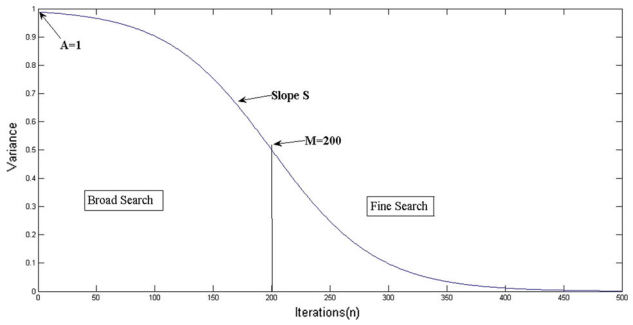

Fig. 1 The variance curve

the following relation:

$$variance(l) = \frac{Ae^{\frac{-l+M}{S}}}{(1+e^{\frac{-l+M}{S}})}, \tag{13}$$

where A is the starting value of the effectiveness of the re-randomization, M is the iteration number corresponding to the midpoint of the slope, and S is the slope.

It can be seen from Fig. 1 that the parameter S controls the slope of the variance curve. The lower the value of S, the steeper the curve is, and the higher the value of S, the flatter the curve is. Using this technique, the search space will be divided into two parts. The first one is a broad search, in which the variance will be large and the particles will be re-randomized far from *gbest* to explore more the search space. In the second part, which is the fine search area, smaller values are assigned to the variance so that the particles will be re-randomized near the new *gbest*, because at the end, we want the solution to converge to the optimal value. These two regions will be separated by a midpoint M, where M will decide on the duration of these broad and fine regions.

To implement the above mechanism, we can adjust the inertia weight of each particle independently at each iteration, based on its new fitness evaluation. If a particle attains a better position, then its inertia weight should be increased or maintained as is. However, if its present fitness is not better than its previous one, then there should be a reduction in its inertia weight [23]. The relation of this adaptive inertial weight, the current inertia weight of the ith particle, is given by

$$w_i(l) = \frac{1}{(1+e^{\frac{-\Delta J_i(l)}{s}})}, \tag{14}$$

where $\Delta J_i(l)$ is the difference of the fitness value of the particle from its previous one, and s is used to control the expected fitness range.

The values of Equation (14) will be in the range (0,1), with 0.5 as the midpoint. If the previous fitness level is better, then it will assign a value less than 0.5, meaning that the speed of the particle should be reduced as the new position is not better than the previous one. Otherwise, it will assign a value greater than 0.5, which means that the newly calculated position is better than the previous one, therefore, the speed of the particle should be increased.

The three methodologies stated earlier will be adopted here in such a way that at the beginning of the iterations, the particles will converge swiftly. The adaptive inertia weight technique works well for better convergence rate, and re-randomization of the particles around *gbest*, usually slows down the convergence rate. Therefore, at the beginning, since stagnation will not occur, re-randomization (which is used to avoid stagnation) will not be activated. For the proposed hybrid PSO algorithm, adaptive inertia weight assignment will be effective right from the start of the iterations along with the second technique of enhanced socialized effect through *lbest*. After a number of iterations, re-randomization will be activated in the simulation, and this re-randomization will be applied through the variance curve. Therefore, there is a very good chance that there will be a reasonable improvement in the steady-state error with a good convergence rate.

The position update equation will remain the same as in (6) while the velocity update equation is now defined by the newly proposed recursion given by

$$\begin{aligned} v_{id} &= w_i(l) \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) \\ &\quad + c_2 \times rand() \times \left(p_{gd} - x_{id}\right) \\ &\quad + \sqrt{variance(l)} \times rand() \times \left(p_{lgd} - x_{id}\right), \\ 1 &\le d \le D. \end{aligned} \tag{15}$$

If one compares (15) with (5) and (12), observes the following. Unlike (5) and (12), Equation (15) has an extra term (i.e., $\sqrt{variance(l)} \times rand() \times \left(p_{lgd} - x_{id}\right)$) which speeds up the convergence. This is assessed in the simulation results section. Also, Equation (15) has a variable inertia weight represented by (14). This term is either fixed as in (5) or variable as in (12). Therefore, with the newly above-defined recursion for the velocity of the proposed algorithm, the HPSO will converge faster than the existing PSO-based algorithms.

Finally, Table 1 summarizes the velocity update equations for the different PSO algorithms. As can be seen from this table that our proposed algorithm has four extra multiplications and one extra addition over both PSO algorithms of [4] and [24].

4 Simulation Results

This section reports the simulation results findings for the HPSO algorithm in an equalization setup. First, a performance behavior of the proposed algorithm is compared to those of the popular algorithms in an equalization setup, then

Table 1 Velocity update equations for the different PSO algorithms

Algorithm	Velocity update
PSO-VCF [4]	$v_{id} = k_l \times v_{id} + k_l \times c_1 \times rand() \times (p_{id} - x_{id}) + k_l \times c_2 \times rand() \times (p_{gd} - x_{id}), \quad 1 \le d \le D$
PSO-W [24]	$v_{id} = w \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times rand() \times (p_{gd} - x_{id}), \quad 1 \le d \le D$
Proposed HPSO	$v_{id} = w_i(l) \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times rand() \times (p_{gd} - x_{id}) + \sqrt{variance(l)} \times rand() \times (p_{lgd} - x_{id}), \quad 1 \le d \le D$

the Bit Error Rate (BER) analysis is performed to compare the proposed HPSO algorithm with that of the LMS algorithm.

4.1 Adaptive Equalization Using the HPSO Algorithm

In this section, the performance of the proposed HPSO algorithm will be tested on an adaptive channel equalization scenario. The proposed algorithm will be compared with the main PSO-based algorithms and the LMS algorithm. The digital message applied to the channel is a binary sequence of uniformly distributed random numbers taking on the values $(-1, 1)$. The algorithms will be applied on two systems: a linear and a nonlinear one. For the linear system, two linear time-invariant (LTI) channels will be used and described by the following transfer functions: $H_1(z) = 0.2602 + 0.9298z^{-1} + 0.2602z^{-2}$ and $H_2(z) = 0.408 + 0.816z^{-1} + 0.408z^{-2}$. It should be noted here that the second channel has a large eigenvalue spread causing more damage to the signal. With the nonlinear system, the channel described by $H_1(z)$ will be used for the nonlinear as shown in Fig. 2.

For a fair comparison, the following parameters are used for all the algorithms: the number of taps is 9, the signal-to-noise ratio (SNR) is set to 20dB, the window size, N, is 200, the maximum number of iterations, L, is 500, and all the results produced are obtained by averaging 25 trials in Monte Carlo simulations. For the PSO-based algorithms $x_{\min} = -2$ and $x_{\max} = 2$. The step size used for the LMS algorithm is 0.025. Finally, the rest of the parameters are reported in Table 2.

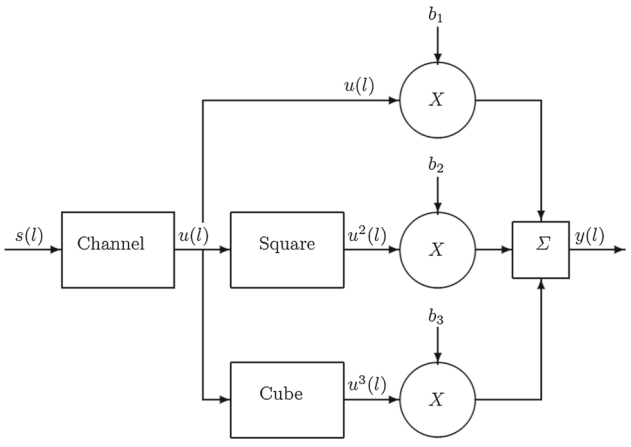

Fig. 2 A nonlinear system where $b_1 = 1$, $b_2 = 0.1$, and $b_3 = 0.05$

Figures 3 and 4 depict the simulation comparison of all the algorithms for the LTI channels given by $H_1(z)$ and $H_2(z)$, respectively. It can be concluded from these two figures that the HPSO exhibits better performance when compared to the rest of the algorithms for both channels. Its consistency in performance is observed on both channels. This is due mainly to the hybridisation action. The three features added to the HSPO algorithm removed all the hurdles facing the other PSO-based algorithms. A 3dB improvement is obtained for both channels. Also, as depicted in Fig. 4, the effect of channel 2 on the performance is more pronounced.

To further investigate the consistency test in performance of the HPSO algorithm, a nonlinear system is used for this purpose. Figure 2 depicts this system where channel 1 is used for this testing. The main reason for using PSO algorithms in nonlinear systems is that they are more effective in such systems compared to other conventional algorithms. Therefore, in order to strengthen this statement, a simulation comparison is carried out and its results are reported in Fig. 5 where all PSO-based algorithms have better performance compared to that of the LMS algorithm, which shows that PSO-based

Table 2 The values of the parameters used in simulation for all algorithms

Parameters	PSO-W	PSO-CCF	PSO-VCF	HPSO
$x_{\min}$	-2	-2	-2	-2
$x_{\max}$	2	2	2	2
Window size N	200	200	200	200
Iteration Number	500	500	500	500
Averaged runs	25	25	25	25
SNR	20 dB	20 dB	20 dB	20 dB
Number of taps	9	9	9	9
$v_{\max}$	$0.07x_{\max}$	$0.2x_{\max}$	$0.2x_{\max}$	$0.09x_{\max}$
$w_{\min}$	0.6	Non	Non	0.6
$w_{\max}$	1	Non	Non	1
$c_1 = c_2$	1.5	4	4	2.5
Particles Number	40	40	Non	30
k	Non	5	Non	Non
$k_{\min}$	Non	Non	4	Non
$k_{\max}$	Non	Non	6	Non
S	Non	Non	Non	45
M	Non	Non	Non	200
A	Non	Non	Non	1

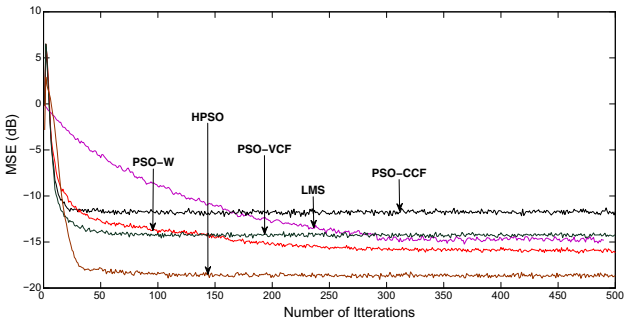

Fig. 3 Convergence behavior of PSO-W, PSO-CCF, PSO-VCF, LMS and HPSO algorithms using $H_1(z)$

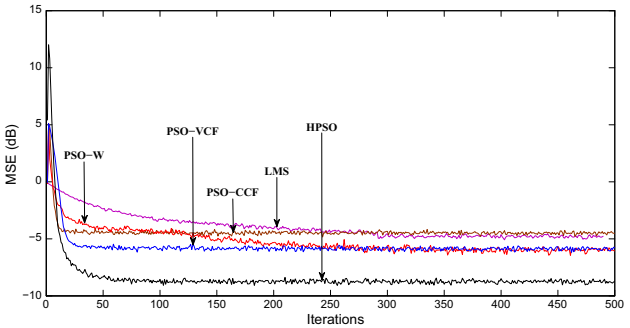

Fig. 4 Convergence behavior of PSO-W, PSO-CCF, PSO-VCF, LMS and HPSO algorithms using $H_2(z)$

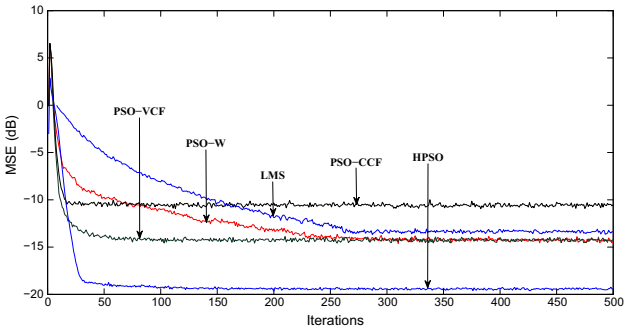

Fig. 5 Convergence behavior of PSO-W, PSO-CCF, PSO-VCF, LMS and HPSO algorithms using the nonlinear system of Figure 2

algorithms have better functionality in nonlinear systems. More importantly, the HPSO algorithm secured an improvement of almost 6dB in steady-state error compared to PSO-W and PSO-VCF. Ultimately, the HPSO secured the minimum steady-state error among all algorithms.

4.2 Bit Error Rate (BER) Analysis

In this section, the performance of the proposed HPSO algorithm is compared with those of the LMS algorithm and the PSO-VCF using the Bit Error Rate (BER) as a metric. The LMS algorithm is the most commonly used algorithm and for this reason is chosen here for this comparison, while the PSO-VCF is chosen as it resulted, except for the HPSO, in the best performance among the rest of the PSO-based algorithms. The BER is obtained for the three algorithms for both the linear system and the nonlinear system described before for the LTI channel given by $H_1(z)$. Figures 6 and 7 show the simulation results of the BER of the three algorithms for the linear system and the nonlinear system, respectively. As expected, the LMS-based equalizer shows a very poor performance, whereas the HPSO equalizer resulted in the best performance.

Figure 6 shows that a BER of 10^{-4} is achieved for an SNR of about 17 dB and 13.5 dB for LMS and HPSO, respectively. Therefore, HPSO outperforms LMS by about 3.5 dB at a BER of 10^{-4}. Figure 7 shows that a BER of 10^{-4} is achieved for an SNR of about 19 dB and 14 dB for LMS and HPSO, respectively. Therefore, HPSO outperforms LMS by about 5 dB at a BER of 10^{-4}. This shows the impact of particle swarm strength over the traditional LMS algorithm.

In conclusion, our proposed HPSO algorithm outperformed both the LMS algorithm and the other PSO-based

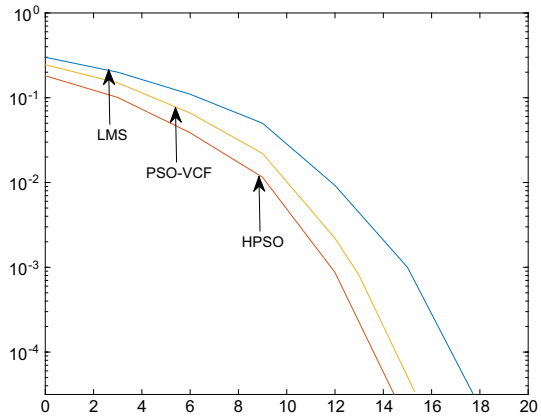

Fig. 6 BER performance of LMS, PSO-VCF and HPSO algorithms for the linear system

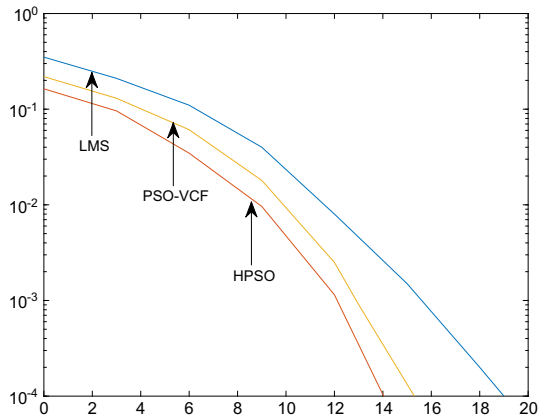

Fig. 7 BER performance of LMS, PSO-VCF and HPSO algorithms for the nonlinear system

algorithms. More importantly, the PSO-based algorithms are in general insensitive to the eigenvalue spread of the channel autocorrelation matrix of the input signal and behave better than the LMS algorithm in nonlinear environments.

5 Conclusion

A new particle swarm optimization algorithm, which is a hybrid PSO (HPSO), is proposed. It incorporates three techniques in order to avoid premature convergence and hence improves the steady-state error. These techniques are: a re-randomization of the particles around *gbest*, an enhanced socialized effect through parameter local best (*lbest*), and an adaptive inertia weight assignment to the particles. The convergence behavior of the HPSO was compared to the most popular PSO-based algorithms such as PSO-W, PSO-CCF, and PSO-VCF plus the LMS algorithm in linear and nonlinear channels. All simulation results show that the proposed HPSO has a better steady-state error when compared to these algorithms. When compared to LMS, the HPSO has a gain of about 4dB and 7dB in linear and nonlinear systems, respectively, and its BER performance is found to be better than that of the LMS algorithm as well.

Acknowledgements The authors thank King Fahd University of Petroleum & Minerals (KFUPM) for sponsoring this work.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no competing interests.

References

1. Qureshi, S.: Adaptive equalization. IEEE Proc. **73**, 1349–1387 (1985)
2. Treichler, J.R.; Larimore, M.G.; Harp, J.C.: Practical blind demodulators for high-order QAM signals. Proc. IEEE **86**, 1907–1926 (1998)
3. Sayed, A.: Adaptive Filters. Wiley, New York (2008)
4. Al-Awami, A.T.; Zerguine, A.; Cheded, L.; Zidouri, A.; Saif, W.: A new modified particle swarm optimization algorithm for adaptive equalization. Dig. Signal Process. **21**(2), 195–207 (2011)
5. Abdelhafiz, A.H.; Hammi, O.; Zerguine, A.; Al-Awami, A.T.; Ghannouchi, F.M.: A PSO based memory polynomial predistorter with embedded dimension estimation. IEEE Trans. Broadcast. **59**(4), 665–673 (2013)
6. Iqbal, N.; Zerguine, A.; Al-Dhahir, N.: Adaptive equalization using particle swarm optimization for uplink SC-FDMA. Electron. Lett. **50**(6), 469–471 (2014)
7. Iqbal, N.; Zerguine, A.; Al-Dhahir, N.: Decision feedback equalization using particle swarm optimization. Signal Process. **108**, 1–12 (2015)
8. Kennedy, J.F.; Eberhart, R.C.; Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)
9. Kennedy, J.; Eberhart, R.: Particle swarm optimization. Proc. IEEE Int. Conf. Neural Networks, 1942–1948 (1995)
10. Shi, Y.; Eberhart, R.C.: Empirical study of particle swarm optimization. Proc. IEEE Int. Congr. Evolut. Comput. **3**, 1945–1950 (1999)
11. Clerc, M.: The swarm and the queen: toward a deterministic and adaptive particle swarm optimization. Proc. IEEE Int. Congr. Evolut. Comput. **3**, 1957 (1999)
12. Shi, Y.; Eberhart, R.C.: Parameter selection in particle swarm optimization, Lecture Notes in Computer Science?Evolutionary Programming VII, vol. 1447, pp. 591-600, (1998)
13. Eberhart, R.C.; Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. Proc. IEEE Int. Congr. Evolut. Comput. **1**, 84–88 (2000)
14. Carlisle, A.; Dozier, G.: An off-the-shelf PSO In: Proc. Workshop On particle Swarm Optimization, Indianapolis, USA (2001)
15. Angeline, P.J.: Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, International Conference on Evolutionary Programming. Springer Berlin Heidelberg, pp. 601–610, (1998)
16. El-Gallad, A.I.; El-Hawary, M.E.; Sallam, A.A.; Kalas, A.: Enhancing the particle swarm optimizer via proper parameters selection. Can. Conf. Electr. Comput. Eng. **2002**, 792–797 (2002)
17. van den Bergh, F.; Engelbrecht, A.P.: Cooperative learning in neural networks using particle swarm optimizers. S. Afr. Comput. J. **26**, 84–90 (2000)
18. van den Bergh, F.; Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**, 225–239 (2004)
19. van den Bergh, F.; Engelbrecht, A.P.: Effect of swarm size on cooperative particle swarm optimizers, Proc. Genetic Evolutionary Computation Conf. (GECCO-2001), pp. 892–899, (2001)
20. Parsopoulos, K.E.; Vrahatis, M.N.: Particle swarm optimization method for constrained optimization problems. Proc. Euro-Int. Symp. Comput. Intell. **2002**, 214–220 (2002)
21. Xu, J.; Xin, Z.: An extended particle swarm optimizer, Proc. 19th IEEE Intl. Parallel and Distributed Processing Symposium, pp. 193–197, (2005)
22. Eberhart, R.C.; Shi, Y.: Particle swarm optimization: developments applications and resources. Proc. IEEE Int. Conf. Evolut. Comput. **1**, 81–86 (2001)
23. Krusienski, D.J.; Jenkins, W.K.: A modified particle swarm optimization algorithm for adaptive filtering. IEEE Int. Symp. Circuits Syst. **111**, 137–140 (2006)
24. Hu, X.; Shi, Y.; Eberhart, R.: Recent advances in particle swarm. Proc. IEEE Congr. Evol. Comput. **1**, 90–97 (2004)
25. Kwong, R.H.; Johnston, E.W.: A variable step-size LMS algorithm. IEEE Trans. Signal Process. **40**(7), 1633–1642 (1992)
26. Eberhart, R.; Kennedy, J.: A new optimizer using particle swarm theory, Proc. 6th IEEE Int. Symp. Micro Mach. Human Sci., pp. 39–43, (1995)