01 Jun 2008

Differential Evolution Particle Swarm Optimization for Digital Filter Design

Bipul Luitel

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Recommended Citation

B. Luitel and G. K. Venayagamoorthy, "Differential Evolution Particle Swarm Optimization for Digital Filter Design," *Proceedings of the IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)*, Institute of Electrical and Electronics Engineers (IEEE), Jun 2008.
The definitive version is available at https://doi.org/10.1109/CEC.2008.4631335

Differential Evolution Particle Swarm Optimization for Digital Filter Design

Bipul Luitel, *Student Member, IEEE* and Ganesh K. Venayagamoorthy, *Senior Member, IEEE*

***Abstract*— In this paper, swarm and evolutionary algorithms have been applied for the design of digital filters. Particle swarm optimization (PSO) and differential evolution particle swarm optimization (DEPSO) have been used here for the design of linear phase finite impulse response (FIR) filters. Two different fitness functions have been studied and experimented, each having its own significance. The first study considers a fitness function based on the passband and stopband ripple, while the second study considers a fitness function based on the mean squared error between the actual and the ideal filter response. DEPSO seems to be promising tool for FIR filter design especially in a dynamic environment where filter coefficients have to be adapted and fast convergence is of importance.**

I. Introduction

A FILTER is a frequency selective circuit that allows a certain frequency to pass while attenuating the others. Filters could be analog or digital. Analog filters use electronic components such as resistor, capacitor, transistor etc. to perform the filtering operations. These are mostly used in communication for noise reduction, video/audio signal enhancement etc. In contrast, digital filters use digital processors which perform mathematical calculations on the sampled values of the signal in order to perform the filter operation. A computer or a dedicated digital signal processor may be used implementing digital filters.

Traditionally, different techniques exist for the design of digital filters. Of these, windowing method is the most popular. In this method, ideal impulse response is multiplied with a window function. There are various kinds of window functions (Butterworth, Chebyshev, Kaiser etc.), depending on the requirements of ripples on the passband and stopband, stopband attenuation and the transition width. These various windows limit the infinite length impulse response of ideal filter into a finite window to design an actual response. But windowing methods do not allow sufficient control of the frequency response in the various frequency bands and other filter parameters such as transition width. Designer always has to compromise on one or the other of the design specifications. So, evolutionary methods have been implemented in the design of digital filters to design with better parameter control and to better approximate the ideal filter. Since population based stochastic search methods have proven to be effective in multidimensional nonlinear environment, all of the constraints of filter design can be effectively taken care of by the use of these algorithms.

Previously, computational Intelligence based techniques such as neural networks, particle swarm optimization (PSO) and genetic algorithms (GA) have been implemented in the design of digital filters. One such approach using neural networks has been described in [1]. Use of PSO and GA in the design of digital filters is described in [2]. Also, use of PSO in the design of frequency sampling finite impulse response (FIR) filter has been described in [3]. Use of differential evolution in the design of digital filters has been implemented in Storn's work [4], [5] and Karaboga's work [6]. Design of infinite impulse response (IIR) filters using PSO is described in [7]. PSO with a preferential velocity update mechanism has been explained and applied in the design of IIR filters in [8].

In this paper, swarm and evolutionary algorithms have been applied for the design of digital filters. Particle swarm optimization and differential evolution particle swarm optimization (DEPSO) have been used here for the design of linear phase FIR filters.

II. Digital Filter Design

Digital filters are classified as finite impulse response (FIR) filter or infinite impulse response (IIR) filter depending upon whether the response of the filter is dependent on only the present input values or on the present inputs as well as previous outputs, respectively.

Any time varying signal $C=x(t)$ sampled at a sampling interval of h has input signals $x_0, x_1, x_2, x_3, \ldots\ldots\ldots, x_n$ in intervals $0, h, 2h, 3h, \ldots\ldots\ldots, nh$. These inputs have corresponding outputs $y_0, y_1, y_2, y_3, \ldots\ldots\ldots, y_n$ depending upon the kind of operation performed. Thus, the order of the filter is determined by the number of the previous input terms used to calculate the current output. The a_0, a_1, a_2 terms appearing in the following equations are called the filter coefficients and determine the operation of the filter.

$$ZeroOrder: y_n = a_0 x_n \quad (1)$$

$$FirstOrder: y_n = a_0 x_n + a_1 x_{n-1} \quad (2)$$

$$SecondOrder: y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} \quad (3)$$

The funding provided by the National Science Foundation, USA under the CAREER grant ECCS #0348221 and ECCS #0625737 is gratefully acknowledged

Bipul Luitel and Ganesh K. Venayagamoorthy are with the Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65401 USA (e-mail: bl7f3@ mst.edu and gkumar@ieee.org).

978-1-4244-1823-7/08/$25.00©2008 IEEE

Now FIR filter or also called the non-recursive filter can be represented by the following equation:

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} \tag{4}$$

Similarly, IIR filter or also called the recursive filter is represented as:

$$y_n = x_n + y_{n-1} \tag{5}$$

By introducing a unit delay element z^{-1}, such that $z^{-1}x_n = x_{(n-1)}$ and $z^{-1}y_n = y_{(n-1)}$, the transfer function of FIR filter can be represented as:

$$\frac{y_n}{x_n} = a_0 + a_1 z^{-1} + a_2 z^{-2} \tag{6}$$

and that of IIR filter as:

$$\frac{y_n}{x_n} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}} \tag{7}$$

Various filter parameters which come into picture are the stopband and passband normalized frequencies (ω_s, ω_p), the passband and stopband ripple (δ_p) and (δ_s), the stopband attenuation and the transition width. These parameters are mainly decided by the filter coefficients as is evident from transfer functions in (6) and (7). Significance of these parameters in actual filters with respect to ideal filter is illustrated in Fig. 1. In any filter design problem, some of these parameters are fixed while others are determined. In this paper, swarm and evolutionary optimization algorithms are applied in order to obtain the actual filter response as close as possible to the ideal response.

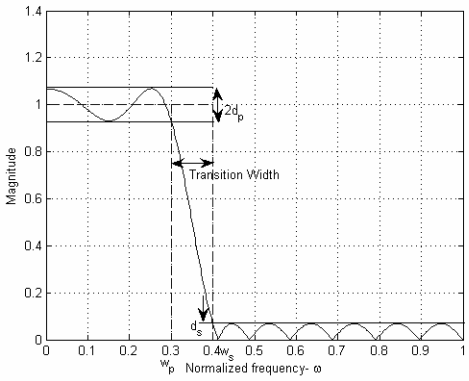

Fig. 1. Ideal and Actual filter Magnitude response showing passband and stopband ripples and transition band in actual filter.

III. DEPSO ALGORITHM

A. Particle Swarm Optimization

Particle swarm optimization is an evolutionary algorithm developed by Eberhart and Kennedy in 1995 [9]. It is a population based search algorithm and is inspired by the observation of natural habits of bird flocking and fish schooling. In PSO, a swarm of particles moves through a D dimensional search space. The particles in the search process are the potential solutions, which move around the defined search space with some velocity until the error is minimized or the solution is reached, as decided by the fitness function. The particles reach to the desired solution by updating their position and velocity according to the PSO equations. In PSO model, each individual is treated as a volume-less particle in the D-dimensional space, with the position and velocity of i^{th} particle represented as:

$$X_i = (x_{i1}, x_{i2}, ..., x_{iD}) \tag{8}$$

$$V_i = (v_{i1}, v_{i2}, ..., v_{iD}) \tag{9}$$

$$V_{id} = w * V_{id} + c_1 * rand_1() * (P_{id} - X_{id}) + c_2 * rand_2() * (P_g - X_{id}) \tag{10}$$

$$X_{id} = X_{id} + V_{id} \tag{11}$$

These particles are randomly distributed over the search space with initial position and velocity. They change their positions and velocity according to (10) and (11) where c_1 and c_2 are cognitive and social acceleration constants, $rand_1()$ and $rand_2()$ are two random functions uniformly distributed in the range of [0,1] and w is the inertia weight introduced to accelerate the convergence speed of PSO [9]. Vector $P_i = (P_{i1}, P_{i2}, ..., P_{iD})$ is the best previous position (the position giving the best fitness value) of particle i called the pbest, and vector $P_g = (P_{g1}, P_{g2}, ..., P_{gD})$ is the position of the best particle among all the particles in the population and is called the gbest. X_{id}, V_{id}, P_{id} are the d^{th} dimension of vector of X_i, V_i, P_i.

The basic pseudocode for PSO can be written as:

For each particle
  Initialize Particle
End
Do
  For each particle
    Calculate Fitness Value
      If the fitness value is better than the best fitness value (pbest) in memory
        Set current value as the new pbest
  End
  Choose the particle with the best fitness value of all the particles as the gbest
  For each particle

Calculate particle velocity using (10)
Calculate particle position using (11)
End
While *Maximum iterations or minimum error criteria is not attained.*

B. Differential Evolution

Differential evolution was introduced by Storn and Price in 1995 [4]. It is yet another population based stochastic search technique for function minimization. Use of DE in the filter design problem has been described in [4] and [5]. In DE, the weighted difference between the two population vectors is added to a third vector and optimized using selection, crossover and mutation operators as in GA. Each individual is first mutated according to the difference operation. This mutated individual, called the offspring, is then recombined with the parent under certain criteria such as crossover rate. Fitness of both the parent and the offspring is then calculated and the offspring is selected for the next generation only if it has a better fitness than the parent [6].

C. Differential Evolution Particle Swarm Optimization

A hybrid of DE and PSO gives a new method of optimization called the differential evolution particle swarm optimization [10]. In DEPSO, new offspring is created by the mutation of the parent. In this paper *gbest* has been taken as the parent and a Gaussian distribution has been considered. For mutation, 4 particles are randomly chosen from the population. The weighted error between these particles' positions is used to mutate the parent and create an offspring [12]. The mutation takes place according to (12).

$$If\ (rand() < CR \quad OR \quad d == k)$$
$$then \quad T_{id} = P_{gd} + \delta_{2,d} \tag{12}$$

$$\delta_{2,d} = \frac{(P_{1,d} - P_{2,d}) + (P_{3,d} - P_{4,d})}{2} \tag{13}$$

where $\delta_{2,d}$ is the weighted error in different dimensions, T_{id} is the offspring and P_{gd} is the *gbest* position of the parent. The mutation takes place under the condition when a random number between [0,1] is less than the reproduction rate *CR* or the particles position in any one randomly chosen dimension, *k,* is mutated. This ensures that offspring is never same as the parent. Then the fitness of the offspring is evaluated and the offspring replaces the parent only if it has a better fitness than the parent, otherwise the parent is retained for the next iteration [10]. Basic flowchart for DEPSO is given in Fig. 2. The pseudocode can be written as follows:

For each particle,
Initialize Particle
End
Do
For each particle
Calculate Fitness Value
If the fitness value is better than the best fitness value (pbest) in memory
Set current value as the new pbest
End If
End
Choose the particle with the best fitness value of all the particles as the gbest
For each particle
Calculate particle velocity using (10)
Calculate particle position using (11)
End
For each particle in parent set,
Select 4 (or 6 for Gaussian distribution) particles
Evaluate weighted difference using equation (6) and (7)
Mutate the parent using equation (5)
End
For each particle in offspring set,
Calculate the fitness
If fitness (parent) < fitness(offspring) then
Replace parent with offspring
End if
End
While *maximum iteration or minimum error criteria not reached.*

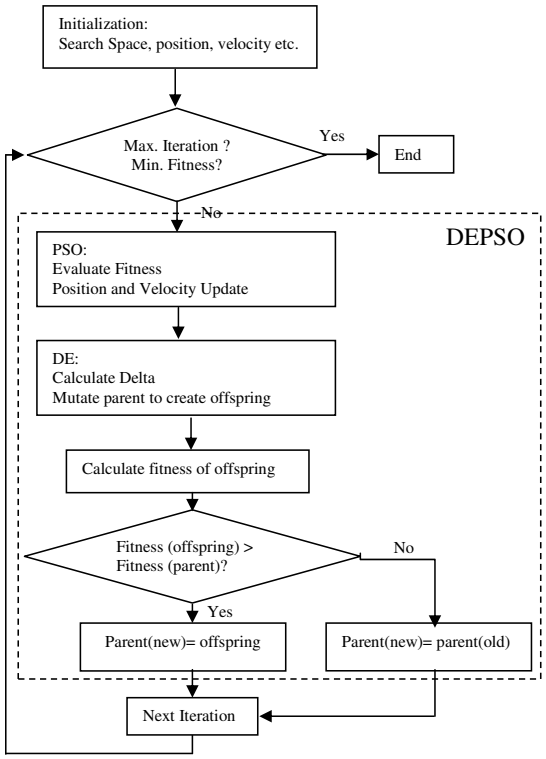

Fig. 2. Basic flowchart showing the three algorithms in the design flow.

A different scheme for mutation in DEPSO is also proposed

in [11], where position update in PSO is carried out either in canonical PSO way or in DE way depending upon the crossover rate.

IV. DEPSO Based Digital Filter Design

From (6), consider an FIR filter with the following transfer function:

$$\frac{y_n}{x_n} = a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n} \qquad (14)$$

or from (7), an IIR filter with the following transfer function:

$$\frac{y_n}{x_n} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n}}{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_m z^{-m}} \qquad (15)$$

Now for (14), the numerator coefficient vector $\{ a_0, a_1, a_2, \ldots\ldots\ldots\ldots, a_N\}$ is represented in N dimensions where as for (15), the numerator as well as denominator coefficient vector is $\{ a_0, a_1, a_2, \ldots\ldots\ldots\ldots, a_N, b_0, b_1, b_2, \ldots\ldots\ldots\ldots, b_M\}$ which is represented in $(N+M)$ dimensions. The particles are distributed in a D dimensional search space, where $D = N$ for FIR and $D = (N+M)$ for IIR filter. The position of the particles in this D dimensional search space represents the coefficients of the transfer function. In each iteration, these particles find a new position, which is the new set of coefficients. Fitness of particles is calculated using the new coefficients. This fitness is used to improve the search in each iteration, and result obtained after a certain number of iterations or after the error is below a certain limit is considered to be the final result. For the problem at hand, the FIR filter is chosen to be of a linear phase type. So its coefficients are matched. Thus the dimension of the problem reduces by a factor of 2. The $D/2$ co-efficient are then flipped and concatenated to find the required D coefficients [2]. Depending on the fitness function used, two different cases have been studied in this paper. The detail operations in the filter design have been summarized in the flowchart in Fig. 3.

A. Case I

Different kinds of fitness functions have been used in different literature. An error function given by (16) is the approximate error used in Parks-McClellan algorithm for filter design.

$$E(w) = G(w)[H_d(e^{jw}) - H(e^{jw})] \qquad (16)$$

Where $G(w)$ is the weighting function used to provide different weights for the approximate errors in different frequency bands, $H_d(e^{jw})$ is the frequency response of the desired filter and $H(e^{jw})$ is the frequency response of the approximate filter [2].

Now the error to be minimized is defined as in (17).

$$J_1 = \underset{w \le w_p}{Max}(|E(w)| - \delta_p) + \underset{w \ge w_s}{Max}(|E(w)| - \delta_s) \qquad (17)$$

where δp and δs are the ripples in the pass and stopband, and w_p and w_s are passband and stopband cut off frequencies respectively. The algorithms try to minimize this error and thus increase the fitness.

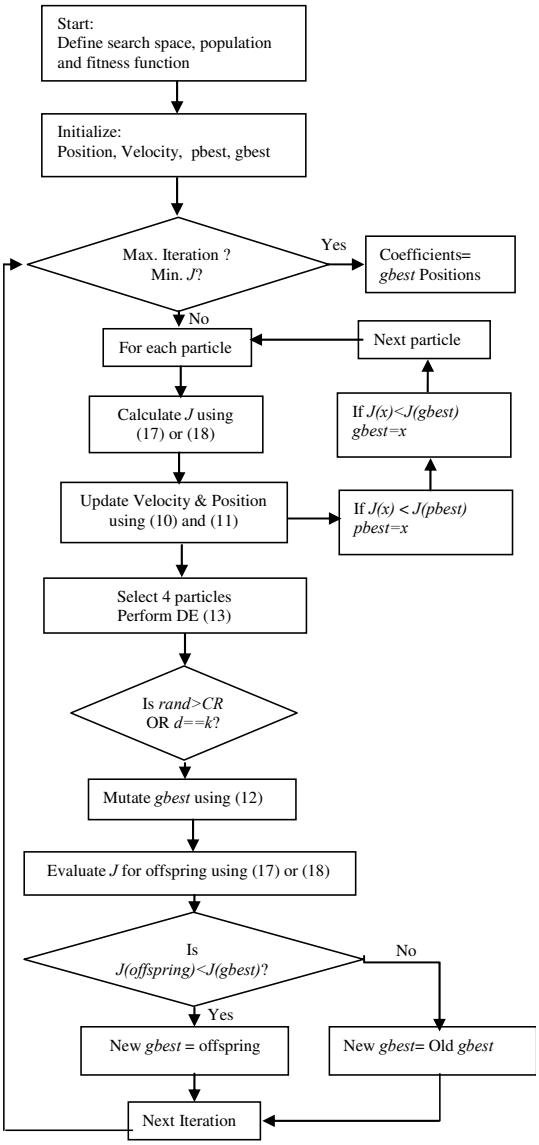

Fig. 3. Detail flowchart showing the design of digital filters using DEPSO algorithm.

B. Case II

The second fitness function takes the mean squared error between the frequency response of the ideal and the actual filter. An ideal filter has a magnitude of 1 on the passband

and a magnitude of 0 on the stopband. So the error for this fitness function is the squared difference between the magnitudes of this filter and the filter designed using the evolutionary algorithms, summed over desired frequency range and divided by the total number of input samples for which the frequency response is evaluated. This is called the mean squared error and is given by (18)

$$J_2 = \frac{1}{N}\sum_{k=1}^{N}(Ideal(k) - Actual(k))^2 \qquad (18)$$

where *ideal(k)* and *actual(k)* are the magnitude response of the ideal and the actual filter, and N is the number of samples used to calculate the error.

V. RESULTS

The filters are designed to optimize the coefficients which give the best frequency response. This is determined by the ripples on the passband and the stopband. In this paper, the desired ripple on the passband δ_p is 0.1 and that on the stopband δ_s is 0.01. In each case, passband and stopband cut off frequencies are 0.25 and 0.3 respectively. The passband ripple is 0.1 and stopband ripple is 0.01. Filters with 20 coefficients are designed. For PSO and DEPSO, population size of 25 is chosen and 200 iterations are used. The value of constants c_1 and c_2 has been taken as 2 and a linearly decreasing w from 0.95 to 0.4 has been used as inertia weight. Gaussian distribution has been used for the DE operation and the operator has been used on the *gbest* for creating offspring. The crossover rate for DEPSO is taken as 0.5. The experiment has been implemented in MATLAB. The coefficients obtained from the filter design have been listed in Table 1.

TABLE I
COEFFICIENTS OBTAINED USING PSO AND DEPSO (200 ITERATIONS)

Coefficients	PSO		DEPSO	
	Case I	Case II	Case I	Case II
1(20)	-0.082247	-0.025779	-0.046829	-0.025779
2(19)	-0.037992	-0.02822	-0.053536	-0.02822
3(18)	-0.018251	-0.007720	-0.042638	-0.007720
4(17)	0.027268	0.027496	0.053059	0.027496
5(16)	0.069825	0.054369	0.045884	0.054369
6(15)	0.045592	0.046422	0.079953	0.046422
7(14)	-0.021912	-0.009960	-0.043754	-0.009960
8(13)	-0.097704	-0.10426	-0.11199	-0.10426
9(12)	0.1962	-0.2034	-0.16226	-0.2034
10(11)	-0.27258	-0.26676	-0.29153	-0.26676

The magnitude and gain plots thus obtained from the design have been plotted. Fig. 4 shows the error graph for Case I. It can be observed that both PSO and DEPSO converge to the same fitness but DEPSO converges to a much lower fitness in lesser number of iterations. The magnitude and gain plot for the filter have been shown in Figs. 5 and 6. Similarly, error curve for Case II has been shown in Fig. 7. Fig. 8 shows the gain plot for Case II. It is seen in Case II that both PSO and DEPSO converge to the same fitness after certain number of iterations and is also evident from the coefficients in Table I. But DEPSO has converged to a much lower error in lesser number of iterations than PSO.

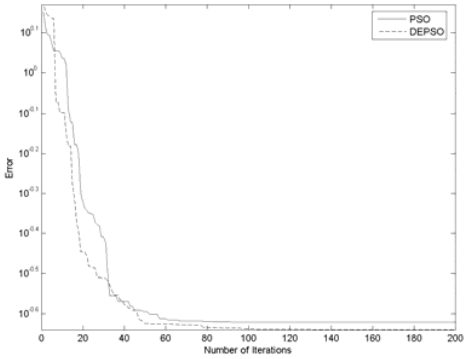

Fig. 4. Error graph for Case I.

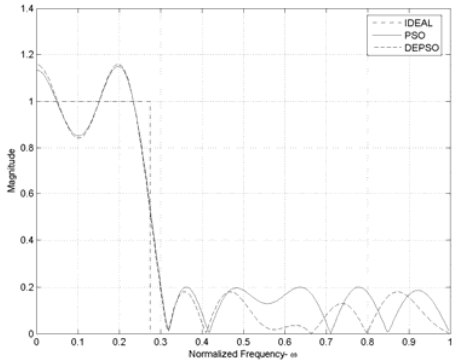

Fig. 5. Magnitude response of filters for Case I (200 Iterations).

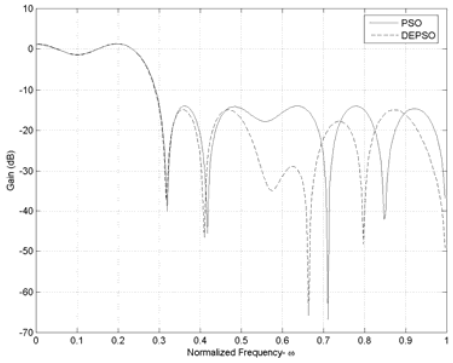

Fig. 6. Gain Plot of the designed filters for Case I (with 200

To show the effectiveness of the two different fitness functions, magnitude and gain for filters designed by each

algorithm using both fitness functions have also been plotted together. Figs. 9 and 10 show the comparison of magnitude and gain plots respectively, using fitness functions in Cases I and II.

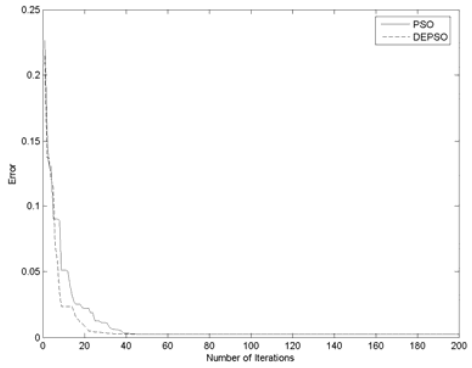

Fig. 7. Error curve for the Case II (with 200 Iterations).

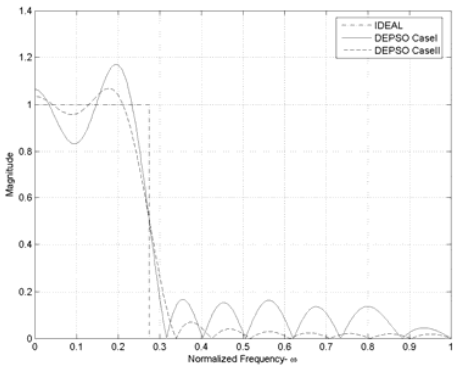

Fig. 9. Comparison of Case I and Case II in terms of magnitude plot.

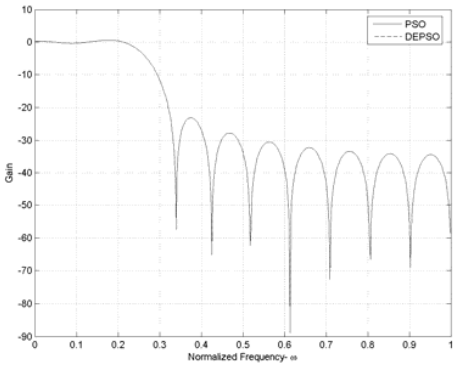

Fig. 8. Gain plot for filter designed in Case II.

TABLE II
PASSBAND AND STOPBAND RIPPLES (WITH 200 ITERATIOINS)

		PSO		DEPSO		Ref [2]
		Case I	Case II	Case I	Case II	
Time	Avg.	9.054	8.899	9.396	9.382	<60
	Min.	8.828	8.796	9.032	9.267	<60
Passband (δ_p)	Avg.	0.174	0.257	0.195	0.257	0.073
	Min.	0.166	0.257	0.169	0.257	0.071
	Max.	0.200	0.257	0.218	0.257	0.075
	Std.	0.009	0.000	0.014	0.000	0.0013
Stopband (δ_s)	Avg.	0.160	0.259	0.182	0.259	0.073
	Min.	0.141	0.259	0.158	0.259	0.071
	Max.	0.185	0.259	0.235	0.259	0.075
	Std.	0.012	0.000	0.025	0.000	0.0013

TABLE III
PASSBAND AND STOPBAND RIPPLES (WITH 40 ITERATIONS)

		PSO		DEPSO		Ref [2]
		Case I	Case II	Case I	Case II	
Time	Avg.	3.108	3.077	4.573	3.015	<60
	Min.	3.021	2.954	3.200	2.875	<60
Passband (δ_p)	Avg.	0.169	0.275	0.172	0.269	0.073
	Min.	0.124	0.256	0.152	0.253	0.071
	Max.	0.266	0.290	0.194	0.291	0.075
	Std.	0.041	0.016	0.018	0.016	0.0013
Stopband (δ_s)	Avg.	0.124	0.263	0.203	0.245	0.073
	Min.	0.190	0.246	0.169	0.207	0.071
	Max.	0.262	0.275	0.257	0.270	0.075
	Std.	0.063	0.012	0.041	0.027	0.0013

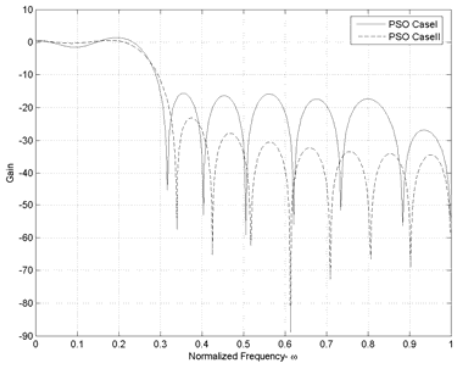

Fig. 10. Comparison of Case I and Case II in terms of gain plot.

The results are evaluated over 10 runs and average, minimum, maximum and standard deviation have been calculated. These compared with the work in [2] have been tabulated in Table II. The results for both the cases run for

40 iterations are shown in Table III. The error graph obtained for Case I in 40 iterations is shown in Fig. 11. The gain and plot for the same case are given in Figs. 12 and 13 respectively.

The use of DEPSO over PSO is thus justified by the results of the experiment run in lesser number of iterations, where DEPSO seems to show better convergence and consistency than PSO. This implies that DEPSO is more suitable for adaptive systems in fast changing environment where quick convergence is the key factor. The gain plot and the error graph for Case II are shown in Figs. 14 and 15 respectively.

VI. Conclusion

Uses of hybrid optimization techniques involve the best practices of both the algorithms and thus help reduce the design time. Also, the fitness is significantly improved because the hybridization helps to save the particles from being trapped in local minima, thus guiding them towards the global solution. From this work, it can be seen that DEPSO performs better with respect to PSO even when its execution time was almost the same as that of PSO. Moreover, if lesser number of iterations be considered, DEPSO would perform much better and which would also mean early approximation of filter coefficients. With fitness function in case study I, transition width was almost always kept within limits while desired ripples were not achieved. But with fitness function in case study II, lower ripples were achieved but at the cost of wider transition width. Since transition width is not the parameter under consideration of this paper, the results in Table II show higher values of ripples than expected.

It can be concluded that with these swarm and evolutionary algorithms, filter approximation can be achieved even in dynamic environments and in a short period of time useful for adaptive filtering. In this paper DEPSO is applied on the *gbest*, but its effectiveness could also be improved by applying it to the *pbest* and thus evaluating the fitness of each particle instead of guiding just one best particle among the population. Further research is required to evaluate these two scenarios for digital filter design especially in a dynamic environment.

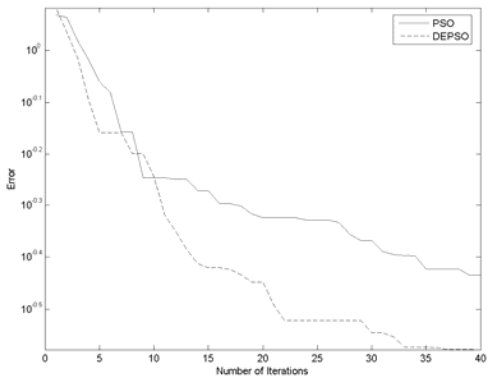

Fig. 11. Error graph for Case I (40 Iterations).

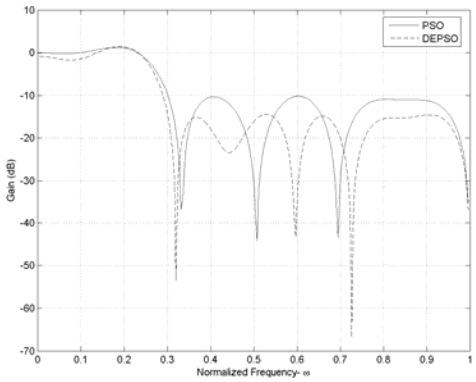

Fig. 12. Gain Plot for Case I (40 Iterations).

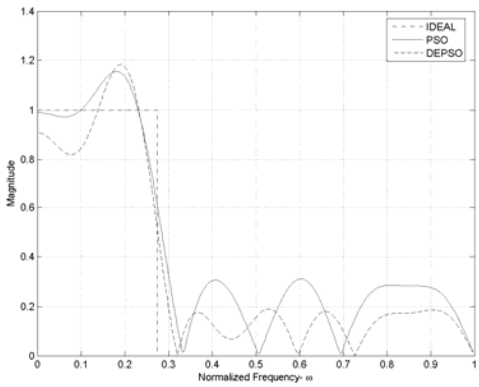

Fig. 13. Magnitude Plot for Case I (40 Iterations).

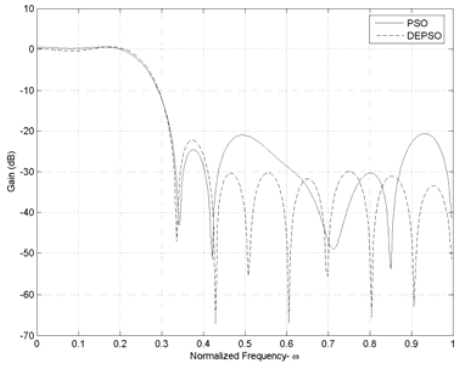

Fig. 14. Gain Plot for Case II (40 Iterations).

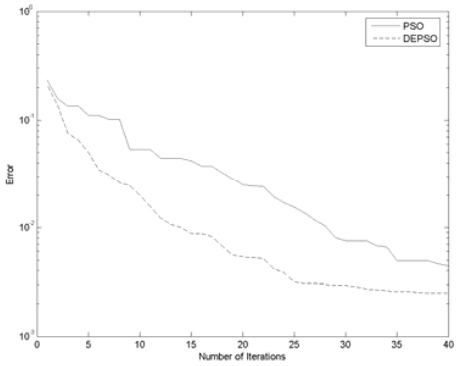

Fig. 15. Error curve for Case II (40 Iterations).

REFERENCES

[1] Wang, X. , Meng X., He, Y., "A Novel Neural Networks-Based Approach for Designing FIR Filters", *The Sixth World Congress on Intelligent Control and Automation*, Vol. 1, 2006, pp. 4029-4032.

[2] Ababneh, J. I., Bataineh M. H., "Linear phase FIR filter design using particle swarm optimization and genetic algorithms," *Digital Signal Process* (2007), doi: 10.1016/j.dsp.2007.05.011.

[3] Wang W. P., Zhou, L. F., Qian, J. X., "Fir Filter Design: Frequency Sampling Filters By Particle Swarm Optimization Algorithm, in Proc. *Of 2004 Int. Conf. Machine Learning and Cybernetics*, vol. 4, Aug. 2004, pp. 2332-2327.

[4] Storn, R., "Differential Evolution Design of an IIR Filter", in *Proc. Of IEEE Int. Conf. Evolutionary Computation*, May 1996, pp. 268-273.

[5] Storn, R., "Designing Nonstandard Filters with Differential Evolution," *Signal Processing Magazine*, *IEEE*, vol. 22, Jan. 2005, pp. 103-106.

[6] Karaboga, N., "Digital Filter Design Using Differential Evolution Algorithm," *EURASIP Journal of Applied Signal Processing,* 2005:8, pp. 1269-1276.

[7] Krusienski, D. J., Jenkins, W. K., "Particle Swarm Optimization for Adaptive IIR Filter Structures," in *Cong. On Evolutionary Computation*, June 2004, vol. 1, pp. 965-970.

[8] Chen, H. C., Chen, O. T., "Particle Swarm Optimization incorporating a Preferential Velocity-Updating Mechanism and Its Application in IIR Filter Design", in *IEEE Int. Conf. Systems, Man and Cybernetics*, Oct. 2006, pp. 1190-1195.

[9] del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., Harley, R. G., "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems", *in IEEE Trans. on Evolutionary Computation*, Accepted for future publication.

[10] Zhang, W., Xie, X., "DEPSO: Hybrid Particle Swarm with Differential Evolution Operator", *in IEEE Int. Conf. Systems, Man and Cybernetics*, Oct. 2003, vol. 4, pp. 3816-3821.

[11] Hao, Z. F, Guo, G. H. , Huang, H., "A Particle Swarm Optimization Algorithm with Differential Evolution", in *Int. Conf. Machine Learning and Cybernetics*, Aug. 2007, Vol. 2, pp. 1031-1035.

[12] Moore, P. W., Venayagamoorthy, G. K., "Evolving Digital Circuits using Hybrid Particle Swarm Optimization and Differential Evolution", *International Journal of Neural Systems,* vol.16, 2006, pp. 1-15.