International Journal of Computational Intelligence Systems, Vol. 6, No. 5 (September, 2013), 911-927

Digital FIR Filter Design Using Hybrid Random Particle Swarm Optimization with Differential Evolution

Vasundhara
Department of Electronics and Communication Engineering
National Institute of Technology Durgapur, West Bengal, INDIA
vdhara2@gmail.com

Durbadal Mandal
Department of Electronics and Communication Engineering
National Institute of Technology Durgapur, West Bengal, INDIA
durbadal.bittu@gmail.com

Sakti Prasad Ghoshal
Department of Electrical Engineering
National Institute of Technology Durgapur, West Bengal, INDIA
spghoshalnitdgp@gmail.com,

Rajib Kar
Department of Electronics and Communication Engineering
National Institute of Technology Durgapur, West Bengal, INDIA
rajibkarece@gmail.com

Received 8 June 2012
Accepted 9 January 2013

Abstract

This paper presents a novel approach of designing linear phase FIR low pass and high pass filter using Random PSO in hybrid with DE known as Random PSODE (RPSODE). In this paper, the Random PSO is used which utilises the weighted particle to guide the search direction for both explorative and exploitative searches. Differential evolution (DE) is one of the very fast and robust evolutionary algorithms which has shown superior performance for continuous global optimization; uses differential information to guide its search direction but sometime causes instability problem; whereas, PSO is a robust, population based stochastic search technique but has the problem of sub-optimality . This paper efficiently combines the Random PSO and DE so as to overcome the disadvantages faced by both the algorithms individually and is used for the design of linear phase low pass and high pass FIR filters. The simulation results show the superiority of RPSODE in global convergence properties and local search ability, and prove it to be a promising candidate for designing the FIR filters. RPSODE outperforms PSO, DE, and PSODE not only in magnitude response but in the convergence speed as well.

Keywords: FIR Filter; PSO; DE; PSODE; Random PSODE; Evolutionary Optimization Technique; Magnitude Response; Convergence.

1. Introduction

Digital Signal Processing (DSP) affords greater flexibility, higher performance (in terms of attenuation and selectivity), better time and environment stability and lower equipment production costs than traditional analog techniques. Additionally, more and more microprocessor

circuitry is being replaced with the cost effective Digital Signal Processing (DSP) techniques and products. A digital filter is simply a discrete-time, discrete-amplitude convolver. In a nutshell, filtering is the multiplication of the signal spectrum by the frequency domain impulse response of the filter. A digital filter computes a quantized time-domain representation of the convolution of the sampled input time function and a representation of the weighting function of the digital filter. They are realized by an extended sequence of multiplications and additions carried out at a uniform spaced sample interval. One can design frequency selective filters, that pass signals with frequency components in some bands while attenuate signals containing frequency components in other frequency bands[1,2]. In a wide sense, digital filter can be classified in two categories: analog and digital filter. Analog filters consist of electronic components operating on continuous time analog signals. Analog filters are greatly affected by the non-linearity of their electronic components, which also drift their values with temperature. No such linearity offset is encountered in digital filter. Digital filter outperforms the analog filter in both roll-off and stop band attenuation. While comparing the magnitude responses of both analog and digital filters, it can be observed that the analog filter has more ripples than the digital filter in the pass band. Digital filters provide better signal to noise ratio as they do not rely upon the analog components. Digital filters are generally available as finite impulse response (FIR) and infinite impulse response (IIR) filters depending on the type of their impulse responses. Finite Impulse Response (FIR) filter has finite impulse responses, which usually decay to zero in a finite amount of time, whereas, impulse responses of Infinite Impulse Response (IIR) filter never die out theoretically. It is easier to design FIR filters with linear phase characteristic. FIR filter is an attractive choice because of the ease in design and stability. By designing the filter taps to be symmetrical about the centre tap position, the FIR filter can be guaranteed to have linear phase. FIR filters are known to have many desirable features such as guaranteed stability, the possibility of exact linear phase characteristic at all frequencies and digital implementation as non-recursive structures.

In order to design a FIR filter, many methods are available such as window method, frequency sampling method etc. Each method has its own merits and demerits. Different types of windows, such as Kaiser, Blackmann, Hanning and Hamming are available depending on the requirement of the filter specifications to be met. The basic idea of windowing consists of approximating the infinite length impulse response of the ideal filter to a finite window to design an actual response[3-5]. The major drawback of window method is that it does not allow sufficient and precise control of various frequencies like pass band, stop band cut-off frequencies and the transition width. For last few years the works have been done continuously to evolve new methods for the filter design. One of the most frequently used methods is Chebyshev approximation method developed by Parks McClellan (PM)[3]. Later on, a well defined computer program was also developed for the design of FIR Filter[6]. All these approaches are the classical ones and have greater tendency to get struck at local minima as they are highly dependent on their starting solutions. For classical optimization to work, the problem should be having continuous and differentiable objective cost function. Complexity of the algorithm and slow convergence speed also limits the usefulness of the classical optimization techniques. The objective function for the design of optimal digital filters involves accurate control of various parameters of frequency spectrum and is thus highly non-uniform, non-linear, non-differentiable and multimodal in nature. Classical optimization methods cannot optimize such objective functions and cannot converge to the global minimum solution because they have the several disadvantages such as: i) highly sensitive to starting points when the number of solution variables and hence the size of the solution space increase, ii) frequent convergence to local optimum solution or divergence or revisiting the same suboptimal solution, iii) requirement of continuous and differentiable objective cost function (gradient search methods), iv) requirement of the piecewise linear cost approximation (linear programming), and v) problem of convergence and algorithm complexity (non-linear programming).

In order to overcome these drawbacks encountered with the traditional and classical methods of optimization, many heuristics optimization techniques have been developed. Particle swarm optimization (PSO) was introduced by Kennedy and Eberhart[7] in 1995. PSO is based on the simulation of simplified animal's social behaviours such as fish schooling, bird flocking, etc.[8] PSO has been widely used for the design of digital filters.[9-11] The limitations of PSO are that they may be influenced

by parameter convergence[12-13]and stagnation problem. In order to overcome the problem encountered with the traditional PSO, the conventional PSO has been modified with some additional features and used for the design of digital filters efficiently like Quantum PSO (QPSO),[14] PSO with Quantum Infusion (PSO-QI),[15-16] Adaptive inertia weight PSO,17 Craziness PSO (CRPSO),[18-19] Chaotic mutation PSO (CMPSO),[20] Improved Particle swarm Optimization,[21-22] PSO with constriction factor approach.[23]

Differential Evolution (DE) is also a heuristic approach for minimizing non-linear and non-differentiable continuous space function has been presented by Storn and Price in the year 1995.[24] It can be implemented very easily and requires less number of control parameters. DE is also used for the design of FIR filters.[25-26] Some modifications have also been done in the classical DE to increase the performance as DE with reserved genes,[27] Modified Differential Evolution algorithm,[28] and self adaptive differential evolution algorithm.[29] DE algorithm is very much sensitive to the choice of these control parameters. Differential evolution in combination with particle swarm optimization is also used so as to overcome their individual disadvantages.[30-31]Particle Swarm Optimization with Differential Evolution (PSODE) is also used for the design of digital filters and digital circuits.[16,32-33] Another form of genetic algorithm known as digit coded genetic algorithm is used in Ref. 34.The MATLAB-Simulink model is proposed for the implementation of FIR low pass and high pass filters in Ref. 35. PSO has been also used with the DE, so as to merge the merits incurred from both the algorithms and to compensate for the individual's shortcomings known as PSODE.[36]

In order to enhance the performance of PSO, to increase the quality of the solution, convergence speed and to avoid trapping of the solution into local minima, several variants of PSO have been developed.[13-21] Though lot of work has been done in this direction for the improvement of PSO, but still space is left for its further improvement. In order to overcome all these problems, the classical PSO has been slightly modified to introduce a new concept of randomly adapting weights in PSO.[37] This paper proposes a novel optimization algorithm which hybridizes the Random PSO with DE in order to overcome the sub-optimality and stagnation problem as faced by the individual algorithm. The proposed algorithm is called Random Particle Swarm Optimization with Differential Evolution (RPSODE) and is employed for the FIR filters design problems. RPSODE utilises the differential information obtained from DE as well as the memory information obtained from PSO. RPSODE helps the particles to increase the population diversity through the combination of PSO operator with the DE operator.

The rest of the paper is arranged as follows. In section 2, the digital filter design problem is formulated. Section 3 briefly discusses about the different evolutionary algorithms PSO, DE, PSODE, and the RPSODE employed for the FIR filter design problems. Section 4 describes the simulation results obtained by employing PM and the above evolutionary algorithms. Section 5 shows the MATLAB-Simulink model implementation of the LP and HP filters designed by RPSODE. Finally, section VI concludes the paper.

2. Design Formulation

Digital filters basically come in two broad categories, namely Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters, depending on their impulse responses. This paper is focussed to the design of FIR filter. The impulse response of the FIR filter is given as:

$$H(z) = h(0) + h(1)z^{-1} + \ldots + h(N)z^{-N} \quad (1)$$

$$H(z) = \sum_{n=0}^{N} h(n)z^{-N} \quad (2)$$

where h(n) is the impulse response. The difference equation representation is

$$y(n) = h(0)x(n) + h(1)x(n-1) + \ldots + h(N)x(x-N) \quad (3)$$

where N is the order of the filter; (N+1) is the number of impulse response coefficients h(n). The values of h(n) will determine the type of the filter, i.e., low pass (LP), high pass (HP) etc., which are to be determined using the above mentioned optimization algorithms individually. This paper presents the optimal designs of linear phase even symmetric Nth order FIR LP and HP filters. While designing the filters, all the generalised specifications for the filter design like flat pass band, highest stop band attenuation, low distortion and narrow transition width have been considered. In each evolutionary algorithm, the individual / particle or vector represents h(n) elements. In each iteration cycle, these particles are updated. Fitness values (i.e., error fitness values which are explained later) of particles are calculated using the new

coefficients. The iteration is continued till the criteria for maximum iteration is reached or the final error fitness value comes out to be below a specified value. One of the major advantages of designing linear phase FIR filter is that it is symmetrical, due to which the coefficients are symmetrical. Only half of the coefficients are updated by any algorithm and then they are concatenated to form the other half due to the symmetrical nature of FIR filter. Therefore, the dimension of the problem is halved.

The frequency response of the FIR digital filter can be calculated as:

$$H(\omega_k) = \sum_{n=0}^{N} h(n) e^{-j\omega_k n} \tag{4}$$

$$\omega_k = \frac{2\pi k}{N}$$

Where $H(w_k)$ is the fourier transform complex vector. This is the FIR filter frequency response. The frequency in $[0, \pi]$ is sampled with N points.

$$H_d(\omega) = [H_d(\omega_1), H_d(\omega_2), H_d(\omega_3), .. H_d(\omega_N)]^T \tag{5}$$

$$H_i(\omega) = [H_i(\omega_1), H_i(\omega_2), H_i(\omega_3) .., H_i(\omega_N)]^T \tag{6}$$

Where $H_d(w)$ represents approximate magnitude response of the designed filter and H_i represents the magnitude response of the ideal filter and for LP and HP it is given, respectively, as Eq.(7.a)- (7.b).

$$H_i(w) = 1 \quad \text{for} \quad 0 \le \omega \le \omega_c, \quad = 0 \quad \text{otherwise.} \tag{7.a}$$

$$H_i(w) = 0 \quad \text{for} \quad 0 \le \omega \le \omega_c, \quad = 1 \quad \text{otherwise.} \tag{7.b}$$

where ω_c is the cut-off frequency of the LP and HP filters.

The error function used in this paper is mean square approach,[32] given as below:

$$Error = \frac{\left\{ \sum_{i=1}^{K} \left[\left| H_d(e^{jw_i}) \right| - \left| H_i(e^{jw_i}) \right| \right] \right\}^2}{N} \tag{8}$$

The error fitness function given in Eq. 8 represents the generalized fitness function to be minimized using the evolutionary algorithms like conventional PSO, DE, PSODE, and the RPSODE individually. Each algorithm tries to minimize this error fitness and thus optimizes the filter performance.

3. Optimization Techniques Employed

3.1. *Particle Swarm optimization*

3.1.1. Classical PSO:

PSO is a flexible, robust population-based stochastic search or optimization technique with implicit parallelism, which can easily handle with non-differential objective functions, unlike traditional gradient based optimization methods. PSO is less susceptible to getting trapped on local optima unlike GA, Simulated Annealing etc. Eberhart *et al.*[7] developed PSO concept similar to the behaviour of a swarm of birds. PSO is developed through simulation of bird flocking and fish schooling in multidimensional space. Bird flocking optimizes a certain objective function. Each agent / particle vector knows its best value so far (pbest). This information corresponds to personal experiences of each vector. Moreover, each vector knows the best value so far in the group (gbest) among all pbests. Namely, each vector tries to modify its position using the following information: the distance between the current position and the pbest, and the distance between the current position and the gbest.

Mathematically, velocities of the vectors are modified according to the following equation[7]:

$$V_i^{k+1} = CFa \times (w^{k+1} * V_i^k + C_1 * rand_1 * (X^k{}_{pbest\_i} - S_i^k) + C_2 * rand_2 * (X^k_{gbest} - S_i^k)) \tag{9}$$

where V_i^k is the velocity of vector i at iteration k; w is the weighting function; C_j is the weighting factor; C_1 and C_2 are called social and cognitive constants, respectively; $rand_i$ *is* the random number between 0 and 1; S_i^k is the current position of vector i at iteration k; $pbest_i^k$ is the pbest of vector i; $gbest^k$ is the gbest of the group. The first term of Eq.(9) is the previous velocity of the vector. The second and third terms are used to change the velocity of the vector. Without the second and third terms, the vector will keep on ‘‘flying’’ in the same direction until it hits the boundary. The parameter *w* corresponds to a kind of inertia and tries to explore new areas. Here, the vector is termed for the string of real filter coefficients, h(n), where n denotes the

dimension of the vector or the number of filter coefficients (N/2+1) for even, symmetric, linear phase Nth order FIR HP filter to be designed. Normally, C_1=C_2=1.5-2.05 and Constriction Factor (CFa) is given in Eq.10.

$$CFa = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \tag{10}$$

where $\phi = C_1 + C_2$, $\phi > 4$ (11)

For $C_1 = C_2$ =2.05, the computed value of CFa = 0.73. The best values of C_1, C_2, and CFa are found to vary with the design sets. Inertia weight (w^{k+1}) at $(k+1)^{th}$ cycle is as given in Eq. (12).

$$w^{k+1} = w_{\max} - \frac{w_{\max} - w_{\min}}{k_{\max}} \times (k+1) \tag{12}$$

where $w_{\max}$ =1.0; $w_{\min}$ =0.4; $k_{\max}$ = Maximum number of iteration cycles. The searching point / updated vector in the solution space can be modified by Eq.(13).

$$S_i^{k+1} = S_i^k + V_i^{k+1} \tag{13}$$

According to Eq.(13), the position of the particle is updated through iterations. These updated positions of the particles form the probable solutions for the next iteration.

3.1.2. Random PSO:

The traditional PSO faces the problem of selection of control parameters, convergence speed and trapping into local minima. A lot of modifications have been done to develop various improved PSO variants but scope is still there for further improvement in the performance of PSO in terms of its convergence speed and the quality of solution. A new variant of PSO known as Random PSO is given in Ref.37, where the PSO posses the ability of randomly adapting using the weighted particle concept.

Random PSO works with the concept of weighted particle. The position of the weighted particle can be calculated as:

$$X_w = \sum_{i=1}^{P} \overline{C}_{w\_i} X_{pbest\_i} \tag{14}$$

$$\overline{C}_{w\_i} = \frac{\hat{C}_{w\_i}}{\sum_{j=1}^{P} \hat{C}_{w\_j}} \tag{15}$$

$$\hat{C}_{w\_i} = \frac{f_{worst} - f\left(X_{pbest\_i}\right)}{f_{worst} - f_{bset}} \tag{16}$$

where X_w presents the weighted particle, like a centre of gravity position

$\overline{C}_{w\_i}$ =normalized weighting constant

$\hat{C}_{w\_i}$ =Weighting Constants

$f(.)$ presents a fitness value

f_{worst} and f_{best} are the global worst and global best fitness values, respectively.

The weighted particle acts as a guiding factor in the search mechanism of the swarm in the population space. It attracts other particles and the swarm converges to the weighted particle gradually through the iterations. The velocities of the particles are calculated as given in Eq. (9) but the position calculation of the particles gets modified slightly according to a controlling parameter known as adapting ratio Eq.(18).If the random value generated is less than the adapting ratio then the position of the particles will be calculated by using the weighted particle concept, whereas if the value of the random number generated is greater than the adapting ratio, then the position of the particle will be calculated using Eq. (9).

$$S_i^{k+1} = S_i^k + (1-\alpha_i)V_i^{k+1} + \alpha_i * C_3 * rand_3 * \left(X_w - S_i^k\right) + \alpha_i * C_4 * rand_4 * \left(X_{pbest\_i} - S_i^k\right) \tag{17}$$

where $\alpha_i = \begin{matrix} 1, rand < adapting\_ratio \\ 0, rand \geq adapting\_ratio \end{matrix}$ (18)

In this case α_i acts like a binary switch which controls Eq.(18). $C_3$and C_4 are acceleration constants.

The value of the adapting ratio should be chosen carefully as it has direct impact on deciding whether the positions of the particles will be calculated directly or using weighted particle.

3.2. *Differential Evolution (DE):*

The concept of DE was first proposed by Storn and Price in 1995.[22] The crucial idea behind DE algorithm is a scheme for generating trial parameter

vectors and adds the weighted difference between two population vectors to a third one. Like any other evolutionary algorithm, DE algorithm aims at evolving a population of N_P, D-dimensional parameter vectors, so-called individuals, which encode the candidate solutions, i.e,

$$\overrightarrow{X_{i,g}} = \left\{X_{1,i,g}, X_{2,i,g}, \ldots X_{D,i,g}\right\} \quad (19)$$

where i = 1, 2, 3,…, N_P. The initial population (at g=0) should cover the entire search space as much as possible by uniformly randomizing individuals within the search constrained by the prescribed minimum and maximum parameter bounds:

$$\begin{matrix}\vec{X}_{\min} = \left\{x_{1,\min},\ldots,x_{D,\min}\right\} \text{and} \\ \vec{x}_{\max} = \left\{x_{1,\max},\ldots,x_{D,\max}\right\}\end{matrix} \quad (20)$$

For example, the initial value of the jth parameter of the ith vector is

$$x_{j,i,0} = x_{j,\min} + rand(0,1)*\left(x_{j,\max} - x_{j,\min}\right) \quad (21)$$

where j = 1, 2, 3,…,D. Here, D = (N/2 +1), the number of the symmetrical real filter coefficients for the Nth order even, symmetric, linear phase FIR LP and HP filters to be designed. (N+1) is the total number of filter coefficients.

The random number generator, *rand* (0,1), returns a uniformly distributed random number in the range [0,1]. After initialization, DE enters a loop of evolutionary operations: mutation, crossover, and selection.

3.2.1 *Mutation*:

Once initialized, DE mutates and recombines the population to produce new population. For each trial vector $x_{i,g}$ at generation g, its associated mutant vector $\vec{v}_{i,g} = \left\{v_{1,i,g}, v_{2,i,g}, \ldots, v_{D,i,g}\right\}$ can be generated via certain mutation strategy. Five most frequently used mutation strategies in the DE codes are listed as follows :

$$"DE/rand/1": \vec{v}_{i,g} = \vec{x}_{r_1^{'},g} + F\left(\vec{x}_{r_2^{'},g} - \vec{x}_{r_3^{'},g}\right) \quad (22)$$

$$"DE/best/1": \vec{v}_{i,g} = \vec{x}_{best,g} + F\left(\vec{x}_{r_1^{'},g} - \vec{x}_{r_2^{'},g}\right) \quad (23)$$

$$"DE/rand-to-best/1": \vec{v}_{i,g} = \vec{x}_{i,g} + F\left(\vec{x}_{best,g} - \vec{x}_{i,g}\right) + F\left(\vec{x}_{r_1^{'},g} - \vec{x}_{r_2^{'},g}\right) \quad (24)$$

$$"DE/best/2": \vec{v}_{i,g} = \vec{x}_{best,g} + F\left(\vec{x}_{r_1^{'},g} - \vec{x}_{r_2^{'},g}\right) + F\left(\vec{x}_{r_3^{'},g} - \vec{x}_{r_4^{'},g}\right) \quad (25)$$

$$"DE/rand/2": \vec{v}_{i,g} = \vec{x}_{r_1^{'},g} + F\left(\vec{x}_{r_2^{'},g} - \vec{x}_{r_3^{'},g}\right) + F\left(\vec{x}_{r_4^{'},g} - \vec{x}_{r_5^{'},g}\right) \quad (26)$$

The indices $r_1^{'}$, $r_2^{'}$, $r_3^{'}$, $r_4^{'}$, $r_5^{'}$ are mutually exclusive integers randomly chosen from the range [1, N_P], and all are different from the base index i. These indices are randomly generated once for each mutant vector. The scaling factor F is a positive control parameter for scaling the difference vector. $x_{best,g}$ is the best individual vector with the best fitness value in the population at generation 'g'. In this paper, (19) has been adopted as the mutation strategy.

3.2.2 *Crossover:*

To complement the differential mutation search strategy, crossover operation is applied to increase the potential diversity of the population. The mutant vector $v_{i,g}$ exchanges its components with the target vector $x_{i,g}$ to generate a trial vector:

$$\vec{u}_{i,g} = \left\{u_{1,i,g}, u_{2,i,g}, \ldots, u_{D,i,g}\right\} \quad (27)$$

In the basic version, DE employs the binomial (uniform) crossover defined as

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \left(rand_{i,j}(0,1) \le C_r \text{ or } j = j_{rand}\right) \\ x_{i,j,g} & \text{otherwise} \end{cases} \quad (28)$$

where j=1, 2,…,D.

The crossover rate C_r is user-specified constant within the range [1, 0], which controls the fraction of parameter values copied from the mutant vector. The parameter j_{rand} is a randomly chosen integer in the range [1, D]. The binomial crossover operator copies the j^{th} parameter of the mutant vector $\vec{v}_{i,g}$ to the corresponding element in the trial vector $u_{i,g}$ if $rand_{i,j}(0,1) \le C_r$ or $j = j_{rand}$. Otherwise, it is copied from the corresponding target vector $\vec{X}_{i,g}$

3.2.3 *Selection*:

To keep the population size constant over subsequent generations, the next step of the algorithm calls for the selection to determine whether the target or the trial vector survives to the next generation i.e., at g=g+1. The selection operation is described in Eq.(29).

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g} & \text{if } f\left(\vec{u}_{i,g}\right) \le f\left(\vec{x}_{i,g}\right) \\ \vec{x}_{i,g} & \text{otherwise} \end{cases} \quad (29)$$

Where f (x) is the objective / error fitness function to be minimized. So, if the new vector yields an equal or lower value of the objective function, it replaces the corresponding target vector in the next generation; otherwise the target is retained in the population. Hence the population either gets better (with respect to the minimization of the objective function) or remains the same in fitness status, but never deteriorates.

The above three steps are repeated generation after generation until the maximum number of generations or iteration cycles is reached. The desired optimal filter coefficients h(n) of size (N/2 +1) and finally after concatenation, (N+1) optimal FIR LP and HP filters coefficients are obtained to get the optimal frequency spectrum.

Proper selection of control parameters is very important for algorithm's success and performance. The optimal control parameters are problem-specific. Therefore, the set of control parameters that best fit each problem have to be chosen carefully. Values of F lower than 0.5 may result in premature convergence, while values greater than 1 tend to slow down the convergence speed. Large populations help maintaining diverse individuals, but also slow down convergence speed. In order to avoid premature convergence, F or N_P should be increased or C_r should be decreased. Larger values of F result in larger perturbations and better probabilities to escape from local optima, while lower C_r preserves more diversity in the population, thus avoiding local optima.

3.3. *Particle Swarm Optimization in combination with Differential Evolution* (*PSODE):*

The limitations of PSO and DE are that they may be influenced by parameter convergence and stagnation problem. To overcome the problems associated with DE and PSO; both of them can be merged so as to overcome their individual disadvantages. In this swarm optimization is combined with the evolutionary optimization to get better accuracy and efficiency of the results. The hybrid of PSO and DE is called PSODE.[30] In PSODE new offspring is created by the mutation of the parent. The personal best and the global best of all the swarms are evaluated by PSO and then these all are mutated and undergo crossover to get better results.

Steps of PSODE are as follows:

- First a population is randomly initialised.
- The fitness value of the population is evaluated Eq.(8), best among the fitness is considered to be gbest.
- The velocity and the position of the particles in the search space is evaluated using Eq. (9) & Eq.(13).
- Now the updated population by PSO will be used to employ differential evolution.
- Donor vector is created by mutation using Eq.(24).
- Trial vector is created by cross over process using Eq.(28).
- The fitness value of the trial vector is calculated and the one having the better fitness value among the trial and the target vector is finally taken as the updated population Eq.(29).

3.4. *Random Particle Swarm Optimization in combination with Differential Evolution (RPSODE):*

In Random Particle Swarm Optimization with Differential Evolution (RPSODE), the classical PSO is slightly modified with the additional concept of weighted particle known as Random PSO[37] and used with DE. In conventional PSO, the swarms which are initialised in the population search space, move in the direction of location of particles associated with pbest and gbest. The movement of the swarms in conventional PSO donot take into account the negative experiences provides by the worst fitness values leading to slow convergence speed and the higher probability of the solution to get struck in local minima solution.The weighted particle behaves in a much similar way like a centre of gravity particle. The weighted particle helps to maintain a balance between the global and local search. A better direction of search is provided by this weighted particle and it avoids trapping of the solution in local optima, thus helps

in improving the quality of the solution as well as convergence speed. The weighted particle is calculated by taking into account both the best and worst fitness values encountered so far. By doing so, the weighted particles help the particles of the search space move closer to the best fitness value and far away from the worst fitness value. The weighted particle is generated if random value is lower than the adapting ratio by using the expressions given in Eq. (14), (15) and (16). If the random value generated is less than the adapting ratio, then the new position of the particles will be calculated by using the weighted particle otherwise the positions will be updated by using the conventional PSO, as given in Eq. (17) and (18). Gradually the swarm converges to the weighted particle after much iteration.

The adapting ratio plays a major role in providing control in the direction of search. The value of adapting ratio has to be chosen carefully. The particles tend to adapt randomly and move according to the weighted particle, if the random value generated is less than the adapting ratio. The adapting ratio can be also varied as the search proceeds. The adapting ratio can be set to a higher value (0.5) in the beginning of the search process so as to favour global search or exploration by helping the swarms to adapt randomly. Gradually as the search proceeds, it can be set to a lower value (0.005) so as to favour local search or exploitation.[37]

The particles are first initialised in the search space. Then according to the value of adapting ratio, the positions of the particles are updated by Random PSO. After the positions of the particles are updated, they undergo mutation and crossover. Thus the diversity of population is maintained by using DE to the results obtained after employing Random PSO.

The steps for the RPSODE are written as follows:

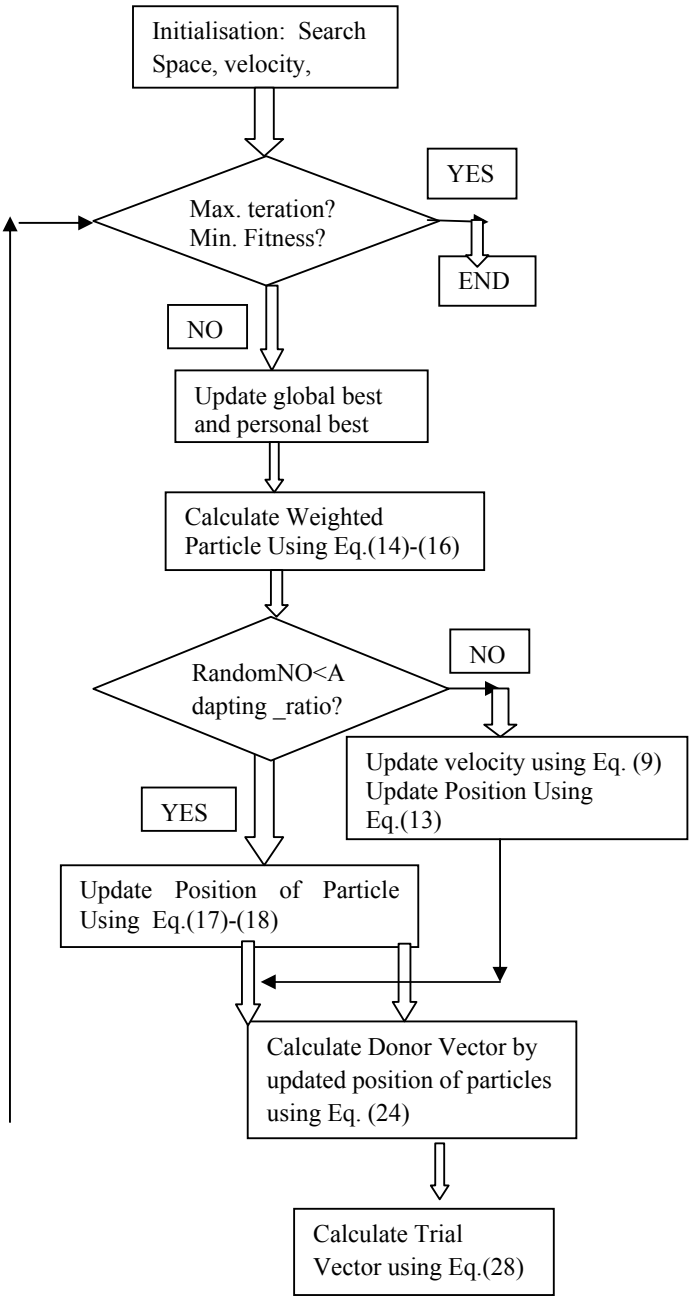
- *Initial Particles:* Randomly generate the initial population matrix.
- *Evaluate Particles:* Calculate the fitness value for each set of particles in the population.
- *Update globalbest:* Take the particle with the best fitness value or the minimum fitness value since it is a minimization problem as the global best.
- *Update Personalbest:* Compare the newly calculated fitness value with the previous one for each and every individual and select the one having the better fitness value as the personal best.
- *Generate a random number to compare with the adapting ratio:* Determine α_i using Eq. (18). If the random number is lesser than the adapting ratio, then α_i is equal to 1 otherwise 0.
- *Calculate weighted Particle:* Calculate the weighted particle using Eq. (14-16).
- *Update position of the Particle:* According to the value of α_i calculated, update the position of the particle either through Eq. (17) or Eq.(9) and Eq.(13).
- *Generate donor vector:* The updated particles are used for creating the donor vector by mutation using Eq. (24).
- *Generate Trial Vector:* The trial vector is created by mutation between donor and target vector using Eq.(28).
- *Selection:* Among trial and target vectors, whichever proves itself to be better candidate in terms of fitness is selected for the final population.
- *Termination:* If the maximum number of iterations is reached or terminating condition is satisfied.

Pseudo code can be written as follows:

```
For each particle,
Initialise Particle
End
Do
 For each particle
    Calculate Fitness Value using Eq.(8)
  If the fitness value is better than the best fitness value (Pbest) in memory
      Set the current value as the new pbest
  End If
End
Choose the particle with the best fitness value of all the particles as the gbest
Choose the particle with the worst fitness value of all the particles as the gworst

Calculate the weighted particle using Eq.(14-16)
Calculate α_i using Eq.(18)
If rand<adapting _ratio
     Calculate particle position using Eq.(17)
Else
      Update particle velocity using Eq. (9)
      Update particle position using Eq. (13)
End If
For each particle in Parent set
    Select four Particles
   Calculate the weighted difference
```

Generate donor vector after the mutation process using Eq.(24)
End
Calculate the trial vector using Eq.(28)
End
For each Particle in the trial vector calculate the fitness value using Eq. (8)
If fitness (trial) <fitness (parent)
Replace the parent with the trial vector
End if
End
While maximum iteration or minimum error criteria is not reached
The fitness based RPSODE can also be summarized in the form flowchart shown in Figure 1.



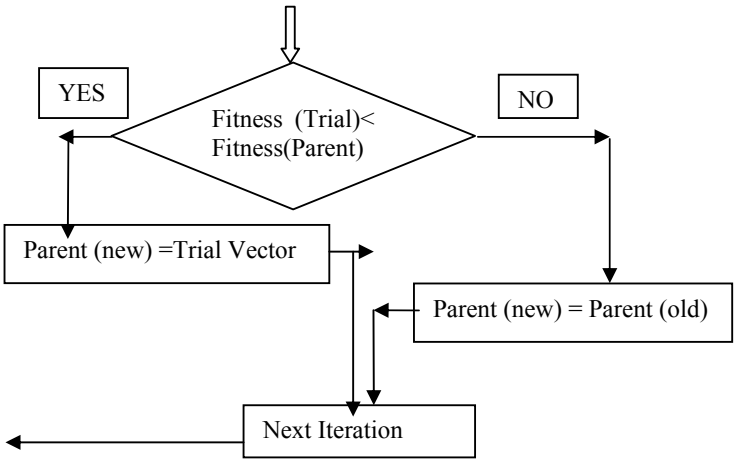


Figure1. Flowchart showing RPSODE.

4. Simulation Results and Discussions

This section presents the extensive simulation works performed on MATLAB for the design of FIR LP and HP filters. For each filter, the filter order is taken as 20, i.e., the number of coefficients is 21. The sampling frequency is taken to be f_s = 1Hz. The number of frequency samples is taken as 128. The algorithms are run for 30 times. Table 1 shows the best chosen parameters for PSO, DE, PSODE, and RPSODE, respectively.

Tables 2-3 show the optimized filter coefficients obtained for FIR LP and HP filters, each of order 20 when PSO, DE, PSODE, and RPSODE are individually adopted. Table 4 shows the comparison of the maximum stop band attenuations achieved for LP and HP filters using PM, PSO, DE, PSODE and RPSODE, respectively. Table 4 shows that the maximum stop band attenuation achieved for the LP filter using the RPSODE is 36.19dB and for the HP filter, the maximum stop band attenuation is 32.82 dB. It is also observed from Table 4 that the RPSODE achieves the best stop band attenuation, as compared to those of PM, PSO, DE, and PSODE for both types of FIR filters.

Table1. PSO, DE, PSODE and RPSODE Parameters

Parameters	PSO	DE	PSODE	RPSODE
Population Size	20	20	20	20
Iteration Cycles	100	100	100	100
Crossover rate	----	0.8	0.8	0.8
Scaling Factor	----	0.7	0.7	0.7
C_1	2.05	----	2.05	2.05
C_2	2.05	----	2.05	2.05
w_{max}	1	----	1	1
w_{min}	0.4	----	0.4	0.4
Adapting_ratio	----	----	----	0.6

Table2. Optimized coefficients of FIR LP filter of order 20 using PSO, DE, PSODE and RPSODE

h(N)	PSO	DE	PSODE	RPSODE
h(1)=h(21)	0.0172	0.0159	0.0214	0.0274
h(2)=h(20)	0.0439	0.0461	0.0478	0.0392
h(3)=h(19)	-0.0022	0.0047	0.0039	0.0032
h(4)=h(18)	-0.0421	-0.0354	-0.0335	-0.0313
h(5)=h(17)	0.0056	0.0008	0.0060	0.0074
h(6)=h(16)	0.0535	0.0589	0.0624	0.0614
h(7)=h(15)	0.0000	0.0050	0.0034	0.0009
h(8)=h(14)	-0.1114	-0.1006	-0.0981	-0.0967
h(9)=h(13)	-0.0041	0.0024	0.0024	0.0038
h(10)=h(12)	0.3076	0.3147	0.3177	0.3154
h (11)	0.4953	0.4999	0.4978	0.4926

Table3. Optimized coefficients of FIR HP filter of order 20 using PSO, DE, PSODE and RPSODE

h(N)	PSO	DE	PSODE	RPSODE
h(1)=h(21)	0.0153	0.0164	0.0251	0.0199
h(2)=h(20)	-0.0447	-0.0437	-0.0394	-0.0419
h(3)=h(19)	0.0017	0.0019	0.0024	0.0052
h(4)=h(18)	0.0386	0.0390	0.0429	0.0345
h(5)=h(17)	0.0001	0.0022	0.0099	0.0030
h(6)=h(16)	-0.0529	-0.0538	-0.0589	-0.0590
h(7)=h(15)	0.0032	-0.0008	0.0033	0.0055
h(8)=h(14)	0.1044	0.1034	0.1069	0.1030
h(9)=h(13)	0.0023	0.0026	-0.0039	-0.0009
h(10)=h(12)	-0.3087	-0.3097	-0.3151	-0.3076
h (11)	0.4878	0.4915	0.4886	0.4955

Table 4.Comparison summary of stop band attenuations for FIR LP and HP filters, each of order 20

Filter Type	Maximum Stop-band Attenuation (dB)				
	PM	PSO	DE	PSODE	RPSODE
LP	23.56	27.06	28.69	29.88	36.19
HP	23.55	26.87	28	29.13	32.82

Table 5 summarizes the comparison of the results obtained by using RPSODE with other reported results. The RPSODE based approach for 20th order LP filter design results in 36.19 dB stop band attenuation and transition width = 0.1093. The results reported in Ref.16, shows that for the LP filter of same order, the maximum stop band attenuation (dB) is less than 27dB (approx) and transition width is more than 0.15.

The simulation results obtained for HP filter using RPSODE are as follows: 32.82 dB stop band attenuation and transition width = 0.1015. It is observed from Table 5 that the simulation results obtained for filter order 20 using RPSODE are much better than the other reported results.

Table 5. Compariosn of RPSODE with other reported results

Model	Parameters			
	Filter type	Order	Maximum stop band attenuation (dB)	Transition width
Karaboga [16]	Low pass	20	NR*	>0.16
Luitel *et al.* [31]	Low Pass	20	<27 dB	>0.13
Najjarzadeh *et al.* [20]	Low Pass	33	<29dB	NR*
Ababneh *et al.* [22]	Low pass	30	<30dB (Approx.)	0.05
Sarangi *et al.* [25]	Low Pass	20	< 27dB	>0.15
RPSODE	Low Pass	20	36.19	0.1093
	High Pass	20	32.82	0.1015

Figure 2 shows the magnitude response of the LP filter using PM, PSO, DE, PSODE, and RPSODE. Figure 3 shows the magnitude response of the HP filter using the same algorithms. From Figures 2-3, it is observed that RPSODE gives the best magnitude response as compared to PSO, DE and PSODE, respectively.

.

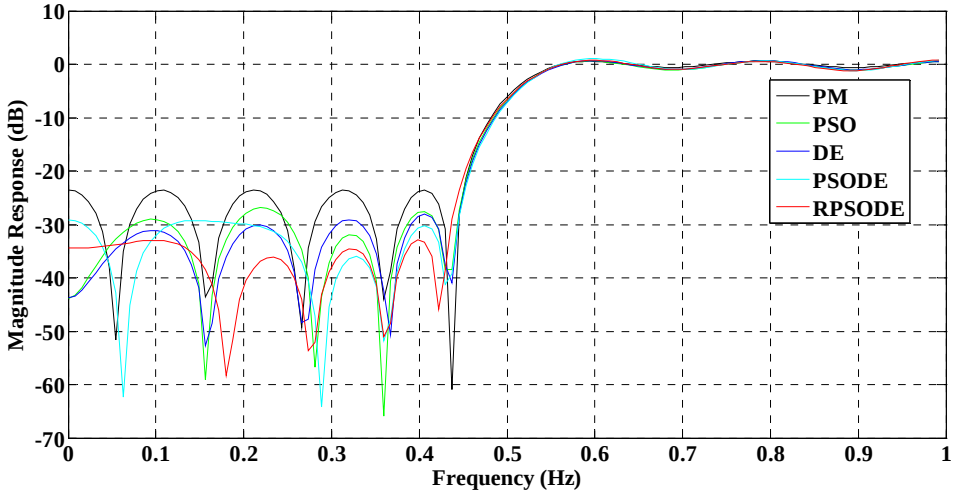


Figure 2: dB plots for the FIR LP filter of order 20.

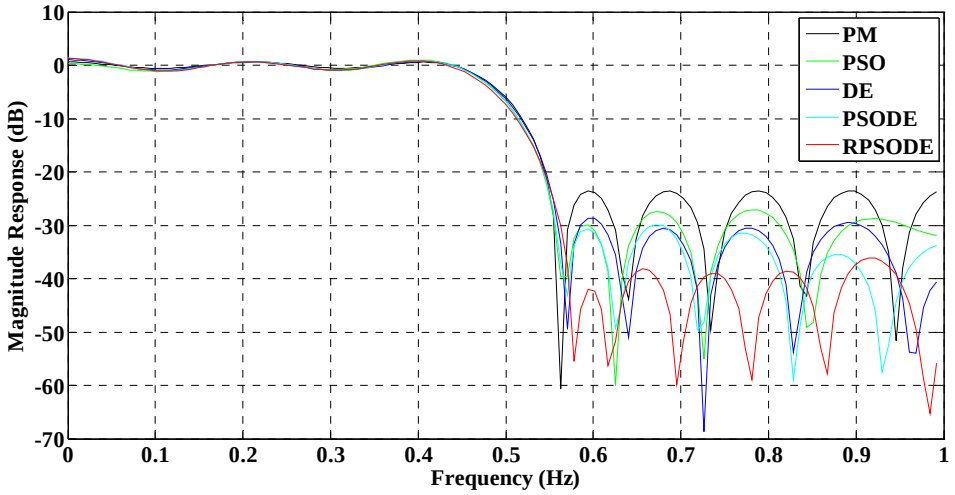


Figure 3: dB plots for the FIR HP filter of order 20.

From the above figures and tables, it is observed that the RPSODE model results in the best magnitude response (dB) and stop band ripple for both LP and HP filters.

4.1. *Comparative effectiveness and convergence profiles of PSO, DE, PSODE and RPSODE algorithms.*

Figure 4 shows the convergence profiles obtained for LP filter using PSO, DE, PSODE and RPSODE algorithms, respectively. The minimum error values

are plotted against the number of iteration cycles to get the convergence profiles for the optimization

From Figure 4, it is observed that RPSODE converges to a much lower fitness value than PSODE, PSO and DE, respectively, for the LP filter. It is seen from Figure 4 and Table 6 that RPSODE converges to a fitness value of 3.005 in 5.2562s, whereas PSODE converges to 3.02 in 5.2374s, DE converges to 3.029 in 3.2284s and PSO converges to 3.039 in 3.4508s, respectively. Figure 5 shows the convergence profiles obtained for the HP filter. It is observed from Figure 5 and Table 6 that RPSODE converges to a fitness value of 3.035 in 5.2746s, whereas PSODE, DE and PSO converge to 3.045 in 5.1093s, 3.086 in 3.4528s and 3.109 in 3.5785s, respectively. Thus for both types of filter RPSODE converges to much lower fitness value than PSODE, PSO and DE, respectively.

Table 6.Comparison of minimum error fitness value and execution time for LP and HP filters

Filter Type	Low Pass Filter (LP)		High Pass Filter (HP)	
	Minimum Error Fitness Value	Execution Time for 100 cycles	Minimum Error Fitness Value	Execution Time for 100 cycles
PSO	3.039	3.4508s	3.109	3.5785s
DE	3.029	3.2284s	3.086	3.4528s
PSODE	3.02	5.2374s	3.045	5.1093s
RPSODE	3.005	5.2562s	3.035	5.2746s

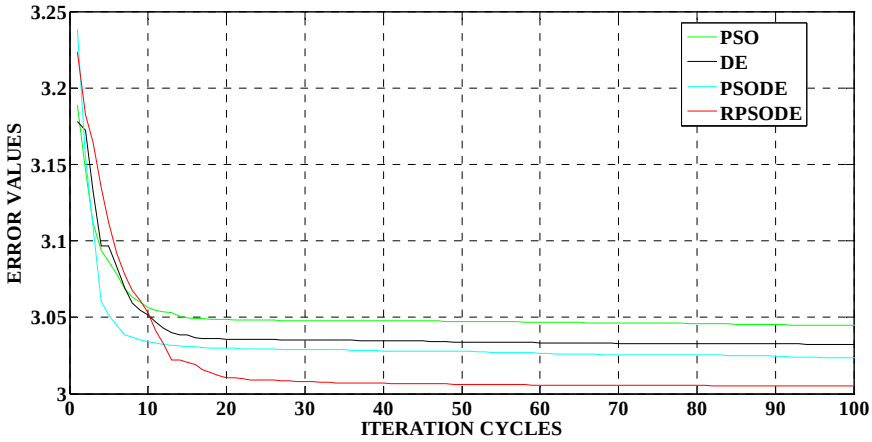


Figure 4. Comparison of convergence profiles for PSO, DE, PSODE and RPSODE of LP filter of order 20.

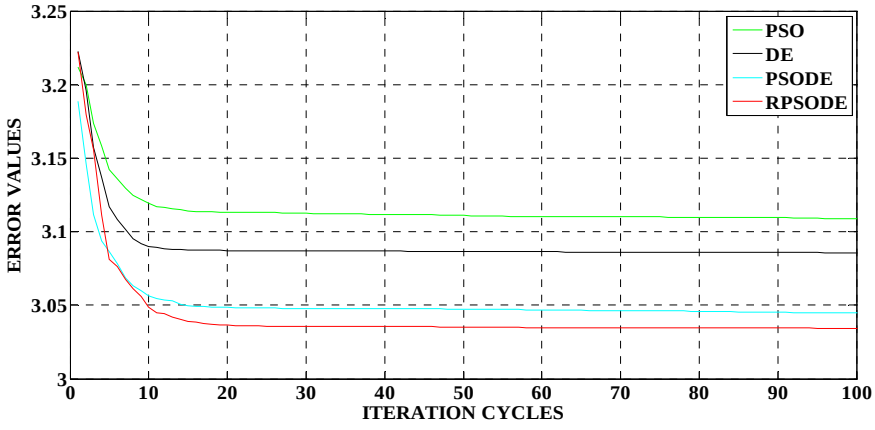


Figure 5. Comparison of convergence profiles for PSO, DE, PSODE and RPSODE of HP filter of order 20.

5. Simulink Model Of The LP and HP Filter Designed By The RPSODE Algorithm.

Figure 6 shows the implementation of the FIR LP and HP filters designed by RPSODE algorithm as a Simulink model in MATLAB. The Simulink model consists of two digital filter blocks, one random noise source, one sine wave source and vector scopes for displaying the signals.

The Simulink model is used to remove the high frequency noise of the sine wave using LP and HP filters designed by RPSODE. A sine wave is generated by DSP sine wave source. The output of vector scope 2 which is a sine wave is shown in Figure 7. The Digital Filter1 block in Figure 6 behaves as the designed HP filter using the RPSODE based optimal filter coefficients of Table 3 and transforms the random noise into a high frequency noise after passing through it, which is shown in Figure 8. The add block acts as an adder for adding the high frequency noise to the sine wave to make it a noisy sine wave which is shown as the output of vector scope 3 in Figure 9. In Figure 6, the Digital Filter 2 block serves the designed LP filter using the RPSODE based optimal filter coefficients of Table 2. The noisy sine wave is then passed through the digital Filter 2 block. The matrix concatenate block in Figure 6 does the work of adding the original sine wave, the noisy sine wave and the filtered sine wave with eliminated high frequency noise.

Thus the output of vector scope 4 is shown in Figure 10, it is observed that a filtered sine wave with eliminated high frequency noise is obtained, when it is passed through the LP filter designed by the RPSODE. Thus the Simulink model in Figure 6 shows the implementation of the LP and HP filters designed by this proposed algorithm RPSODE.

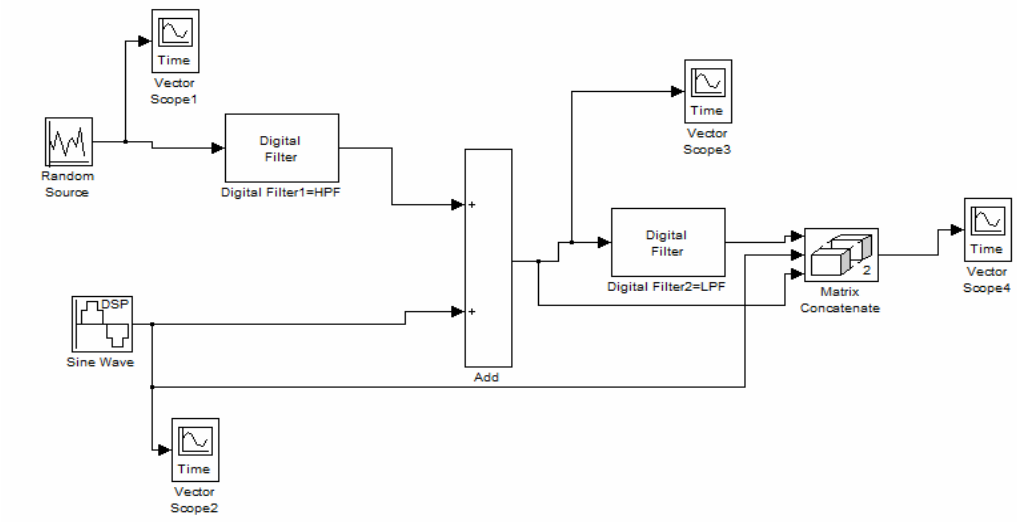


Figure 6. Simulink Model showing the implementation of FIR LP and HP filters designed by RPSODE.

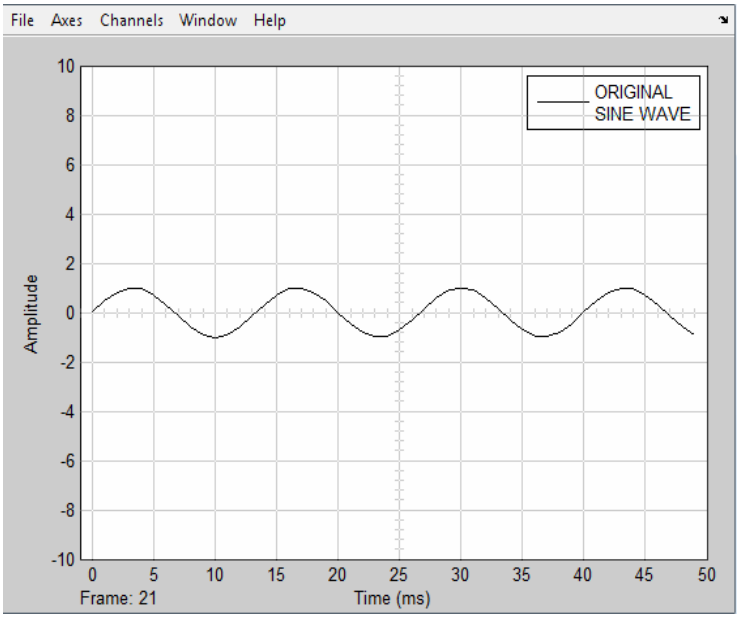


Figure 7. The original sine wave used in the Simulink model.

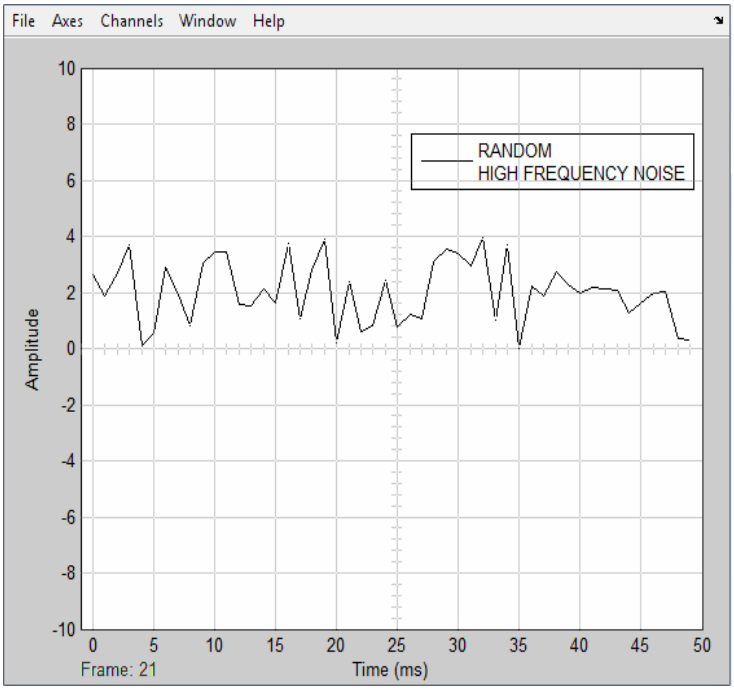


Figure 8. The random high frequency noise used in the Simulink model.

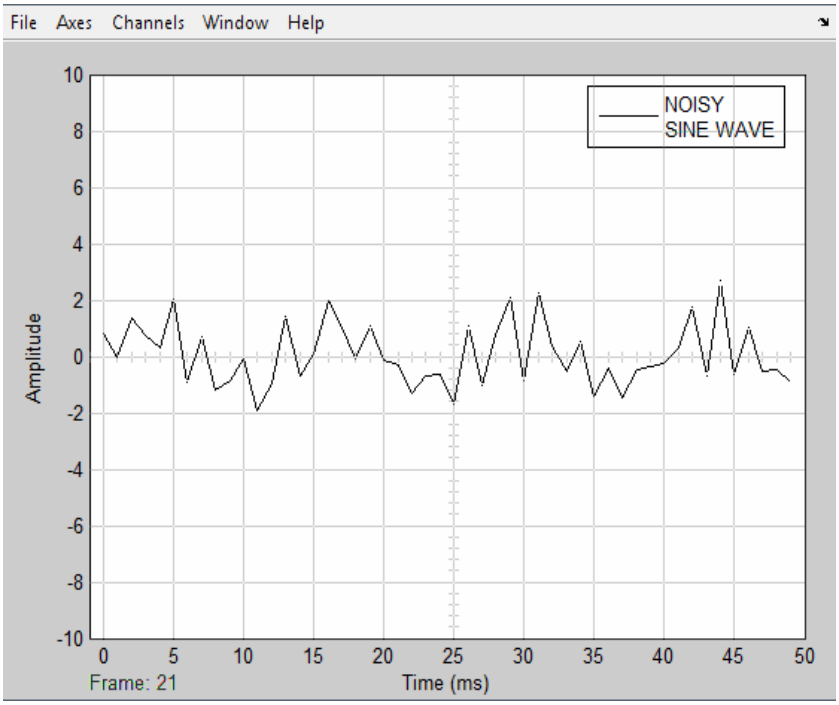


Figure 9. The noisy sine wave used in the Simulink model.

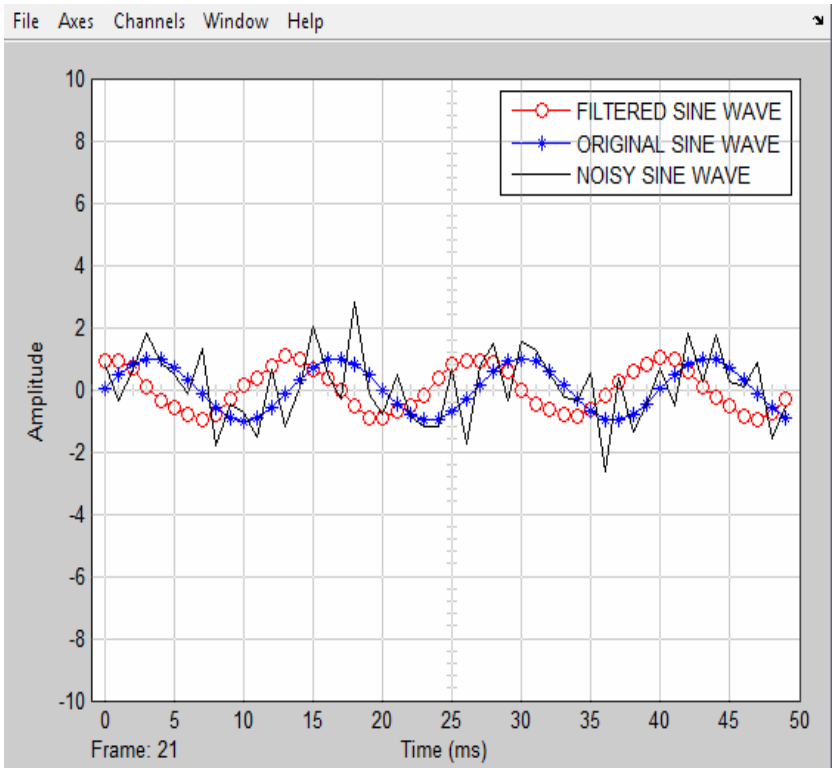


Figure 10. The original sine wave, noisy sine wave and the filtered sine wave used in the Simulink model.

6. Conclusion

In this paper, the authors have proposed an algorithm formed by the combination of Random PSO and DE, termed as Random PSODE (RPSODE). Random PSO algorithm is incorporated into the DE algorithm in order to maintain the diversity and explore the search space more efficiently. Use of hybrid optimization techniques involves the best practices of both algorithms and thus helps to 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. It is revealed that RPSODE has the ability to converge to the best quality near optimal solution and possesses the best convergence characteristics among the algorithms. It is observed from the above given tables and figures that RPSODE is more efficient in successfully optimizing the filter coefficients. RPSODE gives better magnitude response as well as the lowest error value as compared to other algorithms. Thus RPSODE proves itself to be a viable candidate for the optimal design of FIR filters. It is worth noting that, although the algorithm used here is implemented to constrain synthesis for FIR filters, one can see from the proposed technique it is not limited to this case only. It can easily be implemented for the global optimization of Antenna design, Digital filters and in Electrical power systems.

REFERENCES

1. W. Parks, C. S. Burrus, *Digital Filter Design*, Wiley, (New York, 1987).
2. E.C. Ifeachor, B.W. Jervis, *Digital Signal Processing: A Practical Approach*, (Pearson Ed. 2002).
3. T.W. Parks, J.H. McClellan, Chebyshev approximation for non recursive digital filters with linear phase, *IEEE Trans. Circuits Theory CT-19*,189–194, 1972.
4. L.R. Rabiner, Approximate design relationships for low-pass FIR digital filters, *IEEE Trans. Audio Electro acoust. AU-21*, 456–460, 1973.
5. L. Litwin, FIR and IIR digital filters, *IEEE Potentials, 0278-6648*, 2000, pp. 28–31.
6. J.H. McClellan, T.W. Parks, L.R. Rabiner, A computer program for designing optimum FIR linear phase digital filters, *IEEE Trans. Audio Electro. acoust. AU-21,* 1973,pp. 506–526.
7. J. Kennedy and R.C. Eberhart, Particle swarm optimization, *IEEE International Conference on Neural Networks, Piscataway, NJ*, 1995, Vol. 4, pp. 1942-1948.
8. K.E. Parsopoulos and M.N Vrahatis, Recent approaches to global optimization problems through Particle Swarm Optimization, *Natural Computing*, Vo1. 1, pp. 235-306, 2002.
9. M. Najjarzadeh, A. Ayatollahi, FIR Digital Filters Design: Particle Swarm Optimization Utilizing LMS and Minimax Strategies, *IEEE International Symposium on Signal Processing and Information Technology*, 2008. ISSPIT 2008, pp.129-132.
10. D. J. Krusienski, W. K. Jenkins, A modified particle swarm optimization algorithm for adaptive filtering, *IEEE International Symposium on Circuits and Systems*, ISCAS 2006, pp. 137-140.
11. J.I. Ababneh, M.H. Bataineh, Linear phase FIR filter design using particle swarm optimization and genetic algorithms, *Digital Signal Processing*, 2008, 18, (4), 657–668.
12. Ling S. H., Iu H.H.C., Leung F.H.F. and Chan K.Y.: Improved Hybrid Particle Swarm Optimized Wavelet Neural Network for Modelling the Development of Fluid Dispensing for Electronic Packaging, *IEEE Transactions on Industrial Electronics*, vol.55, no.9,September 2008.
13. Biswal B.P., Dash K., Panigrahi B.K.: Power quality disturbance classification using fuzzy C-means algorithm and adaptive particle swarm optimization,*IEEE Trans. Ind. Electron*., Jan. 2009, 56, (1), pp. 162–220.
14. W. Fang, J. Sun, W. Xu, J. Liu, FIR Digital Filters Design Based on Quantum-behaved Particle Swarm Optimization, *First International Conference on Innovative Computing, Information and Control*, ICICIC '06, vol.1, pp.615-619.
15. B. Luitel, G. K. Venayagamoorthy, Particle swarm optimization with quantum infusion for system identification, *Engineering Applications of Artificial Intelligence*, 2010, 23, (5), pp. 635-649.
16. A. Sarangi, R.K. Mahapatra, S.P. Panigrahi, DEPSO and PSO-QI in digital filter design, *Expert Systems with Applications*, 2011, 38, (9), pp.10966-10973
17. X. Yu, J. Liu, H. Li, An Adaptive Inertia Weight Particle Swarm Optimization Algorithm for IIR Digital Filter, *International Conference on Artificial Intelligence and Computational Intelligence (AICI '09),* 2009, vol.1, pp.114-118.
18. S. Mondal, S. P. Ghoshal, R. Kar, D. Mandal, Optimal Linear Phase FIR Band Pass Filter Design using Craziness Based Particle Swarm Optimization Algorithm ,*Journal of Shanghai Jiaotong University (Science), Springer*. Vol. 16, No. 6, 2011, pp. 696-703.
19. S. Mandal, S. P. Ghoshal, R. Kar, D. Mandal, Design of Optimal Linear Phase FIR High Pass Filter using Craziness based Particle Swarm Optimization Technique, *Journal of King Saud*

University – Computer and Information Sciences, Elsevier, vol. 24, pp. 83–92, (2012).

20. Z. Zhao, H. Gao, Y. Liu, Chaotic Particle Swarm Optimization for FIR filter design, *International Conference on Electrical and Control Engineering*, 2011,vol.40, pp2058-2061.
21. S. Mandal, S. P. Ghoshal, R. Kar, D. Mandal, N.V.R. Kishore, FIR Band Stop Filter Optimization by Improved Particle Swarm Optimization, *IEEE World congress on Information and Communication Technologies (WICT) 2011*, pp 699-704.
22. Sangeeta Mondal, Vasundhara, Rajib Kar, Durbadal Mandal, S. P. Ghoshal, Linear Phase High Pass FIR Filter Design using Improved Particle Swarm Optimization, *World Academy of Science, Engineering and Technology 60* 2011.
23. Vasundhara, Rajib Kar, Durbadal Mandal, S. P. Ghoshal, Linear Phase FIR High Pass Filter Design using PSO-CFIWA with Least Square Approach, *2011 IEEE Symposium on Industrial Electronics and Applications (ISIEA2011)*, September 25-28, 2011, Langkawi, pp 314-319.
24. R. Storn, and K. Price, Differential evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces, *Technical Report, International Computer science Institute*, Berkley, 1995.
25. G. Zhao, X. Peng, Design of FIR filters with Differential Evolution, *8th International Conference on Electronic Measurements and Instruments*, 2007, ICEMI'07, pp 748-751.
26. Karaboga N, Cetinkaya1 B. Design of digital FIR filters using differential Evolution algorithm, *Circuits Systems Signal Processing*, 2006, 25, (5), pp. 649–660.
27. G. Liu, Y. Li, G. He, Design of digital FIR filters using differential evolution algorithm based on reserved genes, *IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp 1-7.
28. K.S. Reddy, M.S. Bharath, S.K. Sahoo, S. Sinha, J.P. Reddy, Design of Low Power, High Performance FIR filter using Modified Differential Evolution Algorithm, *International Symposium on Electronic System Design, (ISED)2011*, pp 62-66.
29. S. Chattopadhyay, S. K. Sanyal, A. Chandra, A novel self adaptive differential evolution algorithm for efficient design of multiplier-less low pass FIR filter, *International Conference on Sustainable Energy and Intelligent Systems (SEISCON)* 2011, pp 733-738.
30. W. Zhang, X. Xie, DEPSO: Hybrid Particle Swarm with Differential Evolution Operator, in *IEEE Int. Conf. Systems, Man and Cybernetics*, Oct. 2003, vol. 4, pp. 3816-3821.
31. Z. F. Hao, G. H. Guo, H. Huang, A Particle Swarm Optimization Algorithm with Differential Evolution, *Int. Conf. Machine learning and Cybernetics*, Aug. 2007, Vol. 2, pp. 1031-1035.
32. B. Luitel, G.K. Venayagamoorthy, Differential evolution particle swarm optimization for digital filter design, *IEEE Congress on Evolutionary Computation*, CEC 2008, pp.3954-3961.
33. P. W. Moore, G. K. Venayagamoorthy, Evolving Digital Circuits using Hybrid Particle Swarm Optimization and Differential Evolution, *Int. Journal Of neural Systems*, 2006 Jun, 16(3), pp.163-177.
34. Shing-Tai Pan, A canonic-signed-digit coded genetic algorithm for designing finite impulse response digital filter,*Journal of Digital Signal Processing,vol 20* ,Issue 2,(2010) ,pp-314–327.
35. GAO Jinding, HOU Yubao, SU Long, Design and FPGA Implementation of Linear FIR Low-pass filter Based on Kaiser Window Function, *2011 Fourth International Conference on Intelligent Computation Technology and Automation*,vol.2,(2011),pp 496-498.
36. Xiaochen Wang; Quan Yang; Yuntao Zhao, Research on Hybrid PSODE with Triple Populations Based on Multiple Differential Evolutionary Models, *International Conference On Electrical and Control Engineering (ICECE)*, 2010,pp 1692-1696.
37. Nai-Jen Li, Chen-Chien Hsu, Chih-Min Lin, Global Optimization Using Novel Randomly Adapting Particle Swarm Optimization Approach,*IEEE* 2011.