A swarm intelligence approach to the synthesis of two-dimensional IIR filters

Swagatam Das*, Amit Konar

Abstract

The concept of particle swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient means for intelligent search and optimization. The particle swarm optimization (PSO), as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators and is conceptually very simple. This paper investigates a novel approach to the designing of two-dimensional zero phase infinite impulse response (IIR) digital filters using the PSO algorithm. The design task is reformulated as a constrained minimization problem and is solved by a modified PSO algorithm. Numerical results are presented. The paper also demonstrates the superiority of the proposed design method by comparing it with two recently published filter design methods and two other state of the art optimization techniques.

Keywords: IIR filters; Swarm intelligence; Particle swarm optimization; Differential evolution; Genetic algorithms

1. Introduction

In signal processing, the function of a filter is to remove unwanted parts of a signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range. There are two main kinds of filter, analog and digital. They are quite different in their physical makeup and in how they work. An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op-amps to produce the required filtering effect. Such filter circuits are widely used in applications like noise reduction, video signal enhancement, graphic equalizers in hi-fi systems, and in many other areas. A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialized digital signal processor (DSP) chip.

Digital filters are broadly classified into two main categories namely, finite impulse response (FIR) filters and infinite impulse response (IIR) filters (Oppenheim et al., 1999; Proakis and Manolakis, 1996). An FIR filter is one whose impulse response is of finite duration. The output of such a filter is calculated solely from the current and previous input values. This type of filter is hence said to be non-recursive. On the other hand, an IIR filter is one whose impulse response (theoretically) continues for ever in time. They are also termed as recursive filters. The current output of such a filter depends upon previous output values. These, like the previous input values, are stored in the processor’s memory. The word recursive literally means "running back", and refers to the fact that previously calculated output values go back into the calculation of the latest output. The recursive (previous output) terms feed back energy into the filter input and keep it going.

FIR filters are generally easier to implement, as they are non-recursive and always stable (by definition). On the other hand, it is much more difficult to obtain linear phase responses and to control the overall frequency responses with IIR filters. However, very sharp (narrow transition
Two-dimensional (2D) IIR filters find extensive applications in the domain of denoising of digital images, biomedical imaging and digital mammography, X-rays, image enhancement, seismic data processing, etc. (Kaczorek, 1985; Tzafestas, 1986; Lu and Antoniou, 1992). Multi-dimensional (1D) filter mathematics can also be used in grid methods for solving partial differential equations, distributed control, and iterative learning control.

The most popular design methods for 2D IIR filters are based either on an appropriate transformation of one dimensional (1D) filter (Tzafestas, 1986; Lu and Antoniou, 1992) or on appropriate optimization techniques (Laasko and Ovaska, 1994; Hsieh et al., 1997; Daniel and Willsky, 1997). One of the major problems underlying the design task is to satisfy the stability criterion for the filter transfer function. Although researchers have attempted to tackle the stability problem in a number of ways, most of these efforts (Kaczorek, 1985; Tzafestas, 1986; Lu and Antoniou, 1992; Laasko and Ovaska, 1994; Hsieh et al., 1997) resulted in a filter having a very small stability margin with hardly any practical importance (Mladenov and Mastorakis, 2001). The application of evolutionary computation techniques to the design of digital IIR filters can be traced to the work of Gorne and Schneider (1993). Chellapila et al. (1996) employed an evolutionary programming to optimize the coefficients of the transfer function of a 1D IIR filter. Schiner et al. used an evolutionary multi-objective optimization approach to the design of IIR prototype filters (Kalini and Karaboga, 2005). Very recently, Kalinli et al. proposed an artificial immune system-based methodology for the design of IIR filters (Schnier et al., 2004).

‘Swarm Intelligence’ is the name given to a relatively new interdisciplinary field of research, which has gained a wide popularity in recent times. Algorithms belonging to this field, draw inspiration from the collective intelligence emerging from the behavior of a group of social insects (like bees, termites and wasps). These insects even with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) is a swarm-intelligence-based global optimization technique over continuous search spaces. Since its advent in 1995, PSO has attracted the attention of several researchers all over the world resulting into a huge number of variants of the basic algorithm as well as many parameter automation strategies. To the best of our knowledge, PSO has not been applied to multi-dimensional filter design problem until date.

The contribution of this paper is to present a consistent engineering approach to formal specification-driven optimal design of 2D IIR filters with PSO. In this work, the design task of 2D recursive filters is formulated as a constrained optimization problem. The stability criterion is presented as constraints to the minimization problem. Thus, a generalized filter design framework has been illustrated, which is suitable for application of any global optimization algorithm. We modified the PSO from its classical form (by incorporating a multi-elitist strategy) in order to improve its convergence properties over difficult fitness landscapes. The effectiveness of the modified version has been pointed out in our earlier work reported in Das et al. (2006).

We also employed the differential evolution (DE) (Storn and Price, 1997) and a real coded genetic algorithm (GA) (Goldberg, 1989; Deb et al., 2002a) to the same problem based on this framework. Numerical results show that the PSO-based method yields a better approximation to the transfer function of the filter as compared to the GA or DE-based methods and also beats the state of the art works presented in Mladenov and Mastorakis (2001), Mastorakis et al. (2003) in a statistically significant manner. Compared to the GA-based methods, the PSO algorithm used here is easier to implement and requires fewer function evaluations to find an acceptable solution.

The remainder of this paper is organized as follows. Section 2 provides an overview of the filter design methodology and its reformulation as a constrained minimization problem. In Section 3, the PSO algorithm is presented briefly and its application to the present problem is described in Section 4. Section 5 presents the results of applying the proposed method to a specific design problem and also provides performance comparisons with other optimization techniques. The results are discussed and explained in Section 6 and finally in Section 7 the paper is concluded.

2. Formulation of the design problem

In this work, the filter design is mainly considered from a frequency domain perspective. Frequency domain filtering consists in first, taking the Fourier transform of the 2D signal (which may be the pixel intensity value in case of a gray-scale image), then multiplying the frequency domain signal by the transfer function of the filter and finally inverse transforming the product in order to get the output response of the filter. We illustrate the scheme in Fig. 1.
In this work, we are mainly interested in zero-phase IIR filter design. A zero-phase filter is a special case of a linear-phase filter in which the phase slope is zero. The real impulse response \( h(n) \) of a zero-phase filter is even [i.e. it satisfies \( h(n) = h(-n) \)]. Note that a zero-phase filter cannot be causal. However, in many off-line applications, such as when filtering a sound file or an image file stored on a computer disk, causality is not a requirement and zero-phase filters are usually preferred. Let the general prototype 2D transfer function for the zero-phase IIR digital filter be

\[
H(z_1, z_2) = H_0 \frac{\sum_{i=0}^{N_1} \sum_{j=0}^{N_2} p_{ij} z_1^i z_2^j}{1 + q_k z_1^i + r_k z_1^i + s_k z_1^i z_2^j}
\]  

(1)

The variables \( z_1 \) and \( z_2 \) can be interpreted as complex indecomposable determinants in the discrete Laplace transform (z-transform). The relation of \( z_1 \) and \( z_2 \) with Fourier domain frequency terms \( \omega_1 \) and \( \omega_2 \) are given by

\[
z_1 = e^{j\omega_1}, \quad z_2 = e^{j\omega_2}
\]

(2)

It is a general practice to take \( p_{00} = 1 \) (by normalizing \( p_{ij} \)’s with respect to the value of \( p_{00} \)). Also, let us assume that the user-specified amplitude response of the filter to be designed is \( M_d \), which is obviously a function of digital frequencies \( \omega_1 \) and \( \omega_2 \) \((\omega_1, \omega_2 \in [0, \pi])\). Now the main design problem is to determine the coefficients in the numerator and denominator of Eq. (1) in such a fashion that \( H(z_1, z_2) \) follows the desired response \( M_d(\omega_1, \omega_2) \) as closely as possible. Such an approximation of the desired response can be achieved by minimizing

\[
J(p_{ij}, q_k, r_k, s_k, H_0) = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} \left[ |M(\omega_1, \omega_2)| - M_d(\omega_1, \omega_2) \right]^b,
\]

(3)

where

\[
M(\omega_1, \omega_2) = H(z_1, z_2) \bigg|_{z_1 = e^{j\omega_1}, z_2 = e^{j\omega_2}}
\]

(4)

and

\[
\omega_1 = (\pi/N_1) n_1, \quad \omega_2 = (\pi/N_2) n_2
\]

and \( b \) is an even positive integer (usually \( b = 2 \) or \( 4 \)).

Eq. (3) can be restated as

\[
J = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} \left[ M \left( \frac{\pi n_1}{N_1}, \frac{\pi n_2}{N_2} \right) - M_d \left( \frac{\pi n_1}{N_1}, \frac{\pi n_2}{N_2} \right) \right]^b.
\]

(5)

Here, the prime objective is to reduce the difference between the desired and actual amplitude responses of the filter at \( N_1 \times N_2 \) points. For bounded input bounded output (BIBO) stability the prime requirement is that the z-plane poles of the filter transfer function should lie within the unit circle. Since the denominator contains only first degree factors, we can assert the stability conditions, following (Kaczorek, 1985, Mastorakis et al., 2003; Lu and Antoniou, 1992)

\[
|q_k + r_k| - 1 < s_k < 1 - |q_k - r_k|.
\]

(6)

where \( k = 1, 2, \ldots, N \).

Thus the design of a 2D recursive filter is equivalent to the following constrained minimization problem:

Minimize

\[
J = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} \left[ M \left( \frac{\pi n_1}{N_1}, \frac{\pi n_2}{N_2} \right) - M_d \left( \frac{\pi n_1}{N_1}, \frac{\pi n_2}{N_2} \right) \right]^b
\]

(7a)

subject to the constraints

\[
|q_k + r_k| - 1 < s_k, \quad k = 1, 2, \ldots, N,
\]

\[
s_k < 1 - |q_k - r_k|, \quad k = 1, 2, \ldots, N
\]

(7b)

where \( N_1, N_2 \) and \( N \) are all positive integers. In Mladenov and Mastorakis (2001), the design problem has been tackled with neural networks and the work in Mastorakis et al. (2003) attempts to solve it using a binary coded GA. In the present paper, a much better solution has been obtained using the modified PSO algorithm.

![Fig. 1. Scheme of filtering with 2D digital filter.](image-url)
3. The classical PSO and its modification

PSO (Kennedy and Eberhart, 1995; Kennedy and Eberhart, 2001; Shi and Eberhart, 1998) is in principle such a multi-agent parallel search technique. Particles are conceptual entities, which fly through the multi-dimensional search space. At any particular instant each particle has a position and a velocity. The position vector of a particle is initialized with random positions marked by vectors $\vec{X}_i$ and random velocities $\vec{V}_i$. The population of such particles is called a ‘swarm’ $S$. A neighborhood relation $N$ is defined in the swarm. $N$ determines for any two particles $P_i$ and $P_j$ whether they are neighbors or not. Thus for any particle $P$, a neighborhood can be assigned as $N(P)$, containing all the neighbors of that particle. Different neighborhood topologies and their effect on the swarm performance have been discussed in Kennedy (1999). The PSO used in this work, implicitly uses a so-called fully connected neighborhood topology (or $g_{best}$). Every particle is neighbor of every other particle.

Each particle $P$ has two state variables:

1. Its current position $\vec{x}(t)$
2. Its current velocity $\vec{v}(t)$.

And also a small memory comprising,

1. Its previous best position $\vec{p}(t)$ i.e. personal best experience in terms of the objective function value $f(\vec{p}(t))$.
2. The best $\vec{p}(t)$ of all $P \in N(P)$; $g(t)$ i.e. the best position found so far in the neighborhood of the particle.

The PSO scheme has the following algorithmic parameters:

1. $V_{max}$ or maximum velocity which restricts $\vec{V}_i(t)$ within the interval $[-V_{max}, V_{max}]$.
2. An inertial weight factor $\omega$.
3. Two uniformly distributed random numbers $\varphi_1$ and $\varphi_2$, which respectively determine the influence of $\vec{p}(t)$ and $g(t)$ on the velocity update formula.
4. Two constant multiplier terms $C_1$ and $C_2$ known as “self confidence” and “swarm confidence”, respectively.

Initially the settings for $\vec{p}(t)$ and $g(t)$ are $\vec{p}(0) = g(0) = \vec{x}(0)$ for all particles. Once the particles are initialized, the iterative optimization process begins where the positions and velocities of all the particles are altered by the following recursive equations. The equations are presented for the $d$th dimension of the position and velocity of the $i$th particle:

\[
V_{id}(t + 1) = \omega V_{id}(t) + C_1 \varphi_1 (P_{id}(t) - X_{id}(t)) + C_2 \varphi_2 (g_{id}(t) - X_{id}(t)),
\]

\[
X_{id}(t + 1) = X_{id}(t) + V_{id}(t + 1).
\] (8)

The first term in the velocity updating formula represents the inertial velocity of the particle. The second term involving $p(t)$ represents the personal experience of each particle and is referred to as “cognitive part”. The last term of the same relation is interpreted as the “social term” which represents how an individual particle is influenced by the other members of its society.

In many occasions, the convergence of PSO is premature, especially if the swarm uses a small inertia weight $\omega$ or constriction coefficient (Clerc and Kennedy, 2002). As the global best found early in the searching process may be a poor local minima, we propose a multi-elitist strategy for searching the global best of the PSO. We call the new variant of PSO the MEPSO. The idea draws inspiration from the works reported in Deb et al. (2002b). We define a growth rate $\beta$ for each particle. When the fitness value of a particle of $t$th iteration is higher than that of a particle of $(t-1)$th iteration, the $\beta$ will be increased. After the local best of all particles are decided in each generation, we move the local best, which has higher fitness value than the global best into the candidate area. Then the global best will be replaced by the local best with the highest growth rate $\beta$. Therefore, the fitness value of the new global best is always higher than the old global best. The pseudocode about MEPSO is as follows:

For $t = 1$ to $t_{max}$

For $j = 1$ to $N$/swarm size is $N$

If (the fitness value of particle $j$ in $t$th time-step $> \beta_j$ of particle $j$ in $(t-1)$th time-step)

$\beta_j = \beta_j + 1$;

End

Update local best $j$.

If (the fitness of local best $j$ $> \beta_j$ of global best now)

Choose local best $j$ put into candidate area.

End

End

Calculate $\beta$ of every candidate, and record the candidate of $\beta_{max}$.

Update the global best to become the candidate of $\beta_{max}$.

Else

Update the global best to become the particle of highest fitness value.

End

End

4. Application of the PSO and other evolutionary algorithms to the design problem

4.1. Further simplification of the problem

Without loss of generality let us assume $N = 2$. Then $H$ ($Z_1, Z_2$) in Eq. (1) can be restated as

\[
H(z_1, z_2) = H_\lambda \frac{p_{u_0} z_1 + p_{u_5} z_2 + p_{w_0} z_1 + p_{w_5} z_2 + p_{o_e} z_1 + p_{o_i} z_2 + p_{o_e} z_1 + p_{o_i} z_2 + p_{2} z_1 z_2 + p_{2} z_1 z_2}{(1 + q_1 z_1 + r_1 z_1 + s_1 z_1 z_2)(1 + q_1 z_1 + r_2 z_1 + s_2 z_1 z_2)}.
\] (9)
Fig. 2. Progress towards the optima: Performance comparison of different optimization algorithms on $J_f$ for 50,000 FE.

Now if we substitute $Z_1$ and $Z_2$ as in Eq. (2), then $M (\omega_1, \omega_2)$ can be expressed as

$$M(\omega_1, \omega_2) = H_0 \left[ \frac{p_{00} + p_{01}f_{01} + p_{02}f_{02} + p_{10}f_{10} + p_{20}f_{20} + p_{11}f_{11} + p_{12}f_{12} + p_{21}f_{21} + p_{22}f_{22}}{V} - \frac{f(p_{00} + p_{01}g_{01} + p_{02}g_{02} + p_{10}g_{10} + p_{20}g_{20} + p_{11}g_{11} + p_{12}g_{12} + p_{21}g_{21} + p_{22}g_{22})}{V} \right],$$

(10a)

with $f_{xy} = \cos(x\omega_1 + y\omega_2)$, $g_{xy} = \sin(x\omega_1 + y\omega_2)$

(10b)

and $x, y = 0, 1, 2$

and

$$V = \left[ (1 + q_1f_{10} + r_1f_{01} + s_1f_{11}) - f(q_1g_{10} + r_1g_{01} + s_1g_{11}) \right] \times \left[ (1 + q_2f_{10} + r_2f_{01} + s_2f_{11}) - f(q_2g_{10} + r_2g_{01} + s_2g_{11}) \right].$$

(10c)

From (10a) we may put $M(\omega_1, \omega_2)$ in a compact form as

$$M(\omega_1, \omega_2) = H_0 \frac{N_R - jN_I}{(D_{1R} - jD_{1I})(D_{2R} - jD_{2I})},$$

(11)

where

$$N_R = p_{00} + p_{01}f_{01} + p_{02}f_{02} + p_{10}f_{10} + p_{20}f_{20} + p_{11}f_{11} + p_{12}f_{12} + p_{21}f_{21} + p_{22}f_{22},$$

(12a)

$$N_I = p_{00} + p_{01}g_{01} + p_{02}g_{02} + p_{10}g_{10} + p_{20}g_{20} + p_{11}g_{11} + p_{12}g_{12} + p_{21}g_{21} + p_{22}g_{22},$$

(12b)

$$D_{1R} = 1 + q_1f_{10} + r_1f_{01} + s_1f_{11},$$

(12c)

$$D_{1I} = q_1g_{10} + r_1g_{01} + s_1g_{11},$$

$$D_{2R} = 1 + q_2f_{10} + r_2f_{01} + s_2f_{11},$$

(12d)

$$D_{2I} = q_2g_{10} + r_2g_{01} + s_2g_{11}. $$

(12e)

Hence, the actual magnitude may be written as

$$|M(\omega_1, \omega_2)| = H_0 \sqrt{\frac{(N_R^2 + N_I^2)}{(D_{1R}^2 + D_{1I}^2)(D_{2R}^2 + D_{2I}^2)}}.$$  

(13)

Now let us consider a specific example of the design problem, where the user-specification for the desired circular symmetric low-pass filter response is given as

$$M_d(\omega_1, \omega_2) = 1 \text{ if } \sqrt{\omega_1^2 + \omega_2^2} \leq 0.04$$

$$= 0.5 \text{ if } 0.04 < \sqrt{\omega_1^2 + \omega_2^2} \leq 0.08$$

$$= 0 \text{ otherwise.}$$  

(14)

Also from (5) the constraints may be put in a continuously differentiable form as

$$-(1 + s_k < q_k + r_k < (1 + s_k)$$

$$-(1 - s_k) < q_k + r_k < (1 - s_k)$$

(15)

Now in this problem we experiment with $b = 1, 2, 4$ and $8$, $N_1 = 100$ and $N_2 = 100$. Finally, the constrained minimization task becomes

Minimize

$$f = \left( \sum_{n_i=0}^{100} \sum_{n_i=0}^{100} \left[ |M(\pi n_1, \pi n_2) - M_d(\pi n_1, \pi n_2)|^2 \right] \right),$$

(16a)

subjected to the constraints imposed by (15) with $k = 1, 2$. We define our fitness function as

$$f = \frac{1}{J + \epsilon},$$

(16b)

so that maximization of $f$ leads to minimization of $J$. $\epsilon$ is a small bias term having value 0.001.

4.2. Particle representation

In order to apply the PSO algorithm to the problem formulated in Eq. (16) we need to represent each trial solution as a particle in a multi-dimensional search space. Since $p_{00}$ is always set to 1 in Eq. (1), the dimensionality of the present problem is 14 and each particle has 14 positional coordinates represented by the vector

$$X = (p_{01}, p_{02}, p_{10}, p_{11}, p_{20}, p_{21}, p_{22}, q_1, q_2, r_1, r_2, s_1, s_2, H_0)^T.$$  

(17)
All these 14-dimensional (14D) particles have 14 components in their velocity vector.

4.3. Other competitive algorithms

In 2002, Deb et al. (2002a), proposed a generic parent centric recombination scheme (PCX) and integrated it with a steady state, elite preserving, scalable and computationally fast population alteration model of the GA, which they named the G3 (generalized generation gap) model. Their results indicate that the G3 model with PCX can outperform many other existing GA models and the classical DE when tested on the standard benchmark functions. Below we briefly outline their algorithm. We use the G3 with PCX as our first competitive algorithm. Classical DE (Storn and Price, 1997) has recently gained wide popularity as a fast and efficient optimization algorithm over continuous search spaces. We also compare our proposed algorithm with DE over the zero-phase IIR filter design problem. The particular version of DE that we choose is also known as the ‘DE/rand/1/bin’ since it uses a binary crossover. There are also other frequently used DE operators, such as ‘DE/rand/1/bin’ and ‘DE/best/2/exp’, etc., which we did not investigate in this study.

4.4. Parameter setup for the algorithms

As suggested by Mladenov and Mastorakis (2001) and Mastorakis et al. (2003), we select the initial value of the parameters of the vector in Eq. (17) randomly from the interval (−3, 3). In case of PSO algorithm we vary the inertia weight part linearly from 0.4 to 0.9. We find that this speeds up the convergence of the algorithm to some extent. Other parameter settings and population sizes, used in all experiments reported in this paper, has been shown in Table 1.

4.5. Population initialization and population size

Each space-coordinate of a particle was initialized with a random floating-point number whose absolute value was kept below 3.00. Following Eberhart and Shi (2000), the maximum allowable velocity $V_{\text{max}}$ for each particle was limited to the upper value of the dynamic range of search, i.e., $|V_{\text{max}}| = |X_{\text{max}}| = 3.00$. Eberhart and Shi also showed that the population size has hardly any effect on the performance of the PSO method. It is quite common in PSO research to limit the number of particles in the range 20–60. van den Bergh and Engelbrecht (2001) have shown that though there is a slight improvement of the optimal value with increasing swarm sizes, a larger swarm increases the number of function evaluations necessary to converge to an error limit. We maintain a constant population of 40 particles throughout the runs of the PSO algorithm. For DE we have followed the usual 10 times rule i.e. the number of chromosomes in each generation is kept 10 times the number of dimensions of the chromosomes. Hence, population size for DE is $14 \times 10 = 140$.

4.6. Handling the problem constraints

To handle the constraints we followed the method by Deb (2000) as follows: (a) any feasible solution is preferred to any infeasible solution; (b) between two feasible solutions, the one with a better objective function value is preferred; (c) between two infeasible solutions, the one having a smaller constraint violation is preferred. To tackle the constraints presented in Eq. (15) we start with a population of around 200 particles with randomly initialized positional coordinates. Out of these, 40 particles were selected, space-coordinates of which obey the constraints imposed by (15). If more than 40 particles are initially found to obey the constraints, obviously the selection takes into account the initial fitness value of these particles. During the run of the program, the globally best particle was sorted not only on the basis of its fitness value in the swarm but also depending on whether or not it obeyed the constraints. That is, if a particle in course of its movement through the search space yields the lowest fitness value found so far, its position will be memorized as the globally best position by all other members in the swarm only if it satisfies the constraints. For evolutionary algorithms like DE and GA, only those offsprings were promoted for reproduction or survival to the next generation which satisfies the constraints besides having greater fitness values.

4.7. Filter figures of merit used

The performances of the IIR filters designed here, are evaluated in terms of the following figures of merit:

(1) Error of frequency response in pass band:

$$e_{fp}(\omega) = |H(e^{j\omega}) - H_d(e^{j\omega})| \text{ for } \omega \in [0, \omega_p].$$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>G3 with PCX</td>
<td>MEPSO</td>
<td>DE</td>
<td>GA used in Mastorakis et al. (2003)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pop_size</td>
<td>100</td>
<td>Popsize</td>
<td>40</td>
<td>Popsize</td>
<td>10^4</td>
<td>Pop_size</td>
<td>10^4dim</td>
</tr>
<tr>
<td>$\sigma_l$</td>
<td>0.1</td>
<td>Inertia weight</td>
<td>0.4–0.9</td>
<td>CR</td>
<td>0.9</td>
<td>No. of bits per gene</td>
<td>32</td>
</tr>
<tr>
<td>$\sigma_l$</td>
<td>0.1</td>
<td>$C_1$, $C_2$</td>
<td>1.494</td>
<td>R</td>
<td>0.8</td>
<td>Mutation probability</td>
<td>0.05</td>
</tr>
<tr>
<td>$\mu$</td>
<td>3</td>
<td>$V_{\text{max}}$</td>
<td>3.00</td>
<td>Part of genetic materials interchanged during cross-over</td>
<td>12</td>
<td>Maximum number of children from each pair of parents</td>
<td>10</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(2) Passband magnitude ripple:
\[ e_{\text{pass}}(\omega) = |H(e^{j\omega})| - 1 \] for \( \omega \in [0, \omega_p] \).

(3) Stopband attenuation:
\[ A(\omega) = 20 \log_{10} |H(e^{j\omega})| \] for \( \omega \in [\omega_s, \pi] \).

(4) Maximum magnitude of the poles,

where \( \omega_p \) and \( \omega_s \) are the passband and stopband frequencies, respectively. The transfer function of the filter synthesized by a search algorithm will considerably deviate from the desired transfer function if the error of frequency response in passband is considerably large. Hence, for a good design, the maximum passband error should be as low as possible (Oppenheim et al., 1999; Proakis and Manolakis, 1996). Similarly, the maximum passband ripple should be low enough to ensure a distortionless production of the output signal. On the contrary, the minimum value of passband attenuation should be large in order to ensure the elimination of the undesired frequency components to a high degree.

5. Experiments and results

We run four population-based optimization algorithms namely MEPSO, DE, the G3 with PCX model and the

| Table 2 |
Mean value and standard deviations of the final results (filter coefficients) with exponent \( p = 2 \) after 50,000 FEs (mean of 20 independent runs of each of the competitor algorithms)

<table>
<thead>
<tr>
<th>Filter coefficient</th>
<th>MEPSO</th>
<th>DE</th>
<th>G3 with PCX</th>
<th>GA in Mastorakis et al. (2003)</th>
<th>NN-based method Mladenov and Mastorakis (2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{00} )</td>
<td>0.3061 ± 0.0010</td>
<td>-0.2426 ± 0.002</td>
<td>-0.3016 ± 0.002</td>
<td>1.8193 ± 0.0024</td>
<td>1.8922</td>
</tr>
<tr>
<td>( p_{01} )</td>
<td>0.9949 ± 0.0022</td>
<td>2.4827 ± 0.0063</td>
<td>2.9023 ± 0.0063</td>
<td>-1.1939 ± 0.006</td>
<td>-1.1254</td>
</tr>
<tr>
<td>( p_{10} )</td>
<td>0.3935 ± 0.0068</td>
<td>-0.3484 ± 0.0074</td>
<td>-0.3435 ± 0.0074</td>
<td>0.0725 ± 0.0032</td>
<td>0.0387</td>
</tr>
<tr>
<td>( p_{11} )</td>
<td>-0.0338 ± 0.003</td>
<td>-2.0898 ± 0.001</td>
<td>-2.0490 ± 0.001</td>
<td>-2.5208 ± 0.008</td>
<td>-2.5298</td>
</tr>
<tr>
<td>( p_{12} )</td>
<td>0.6481 ± 0.0067</td>
<td>0.0323 ± 0.0054</td>
<td>0.0387 ± 0.0054</td>
<td>0.2479 ± 0.0022</td>
<td>0.3879</td>
</tr>
<tr>
<td>( p_{20} )</td>
<td>1.2345 ± 0.0021</td>
<td>2.4915 ± 0.0063</td>
<td>2.4932 ± 0.0063</td>
<td>0.6003 ± 0.0024</td>
<td>0.6115</td>
</tr>
<tr>
<td>( p_{21} )</td>
<td>0.5030 ± 0.0037</td>
<td>0.1613 ± 0.0005</td>
<td>0.1975 ± 0.0005</td>
<td>-1.3693 ± 0.005</td>
<td>-1.4619</td>
</tr>
<tr>
<td>( p_{22} )</td>
<td>0.4481 ± 0.0046</td>
<td>0.7564 ± 0.0078</td>
<td>0.7493 ± 0.0078</td>
<td>2.4502 ± 0.0082</td>
<td>2.5206</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>-1.0239 ± 0.081</td>
<td>-0.9113 ± 0.009</td>
<td>-0.4738 ± 0.0009</td>
<td>-0.8603 ± 0.007</td>
<td>-0.8707</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>0.0342 ± 0.0001</td>
<td>-0.0255 ± 0.0366</td>
<td>-0.0843 ± 0.0366</td>
<td>-0.8904 ± 0.006</td>
<td>-0.8729</td>
</tr>
<tr>
<td>( r_1 )</td>
<td>-0.9605 ± 0.035</td>
<td>2.9613 ± 0.0054</td>
<td>2.9493 ± 0.0054</td>
<td>-0.8566 ± 0.003</td>
<td>-0.8705</td>
</tr>
<tr>
<td>( r_2 )</td>
<td>-0.0371 ± 0.002</td>
<td>-0.0344 ± 0.006</td>
<td>-0.0376 ± 0.006</td>
<td>-0.8389 ± 0.006</td>
<td>-0.8732</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>0.9523 ± 0.0078</td>
<td>0.8674 ± 0.0072</td>
<td>0.8874 ± 0.0072</td>
<td>0.7351 ± 0.0052</td>
<td>0.7756</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>-0.9056 ± 0.005</td>
<td>-0.8075 ± 0.0026</td>
<td>-0.8476 ± 0.0026</td>
<td>0.8034 ± 0.0068</td>
<td>0.7799</td>
</tr>
<tr>
<td>( H_0 )</td>
<td>0.00034 ± 0.001</td>
<td>0.0012 ± 0.0073</td>
<td>0.0784 ± 0.0073</td>
<td>0.0093 ± 0.0049</td>
<td>0.0010</td>
</tr>
</tbody>
</table>

| Table 3 |
Mean value and standard deviations of the final results (\( J \) values) with exponent \( p = 1, 2, 4, 8 \) after 50,000 FEs (mean of 20 independent runs of each of the competitor algorithms)

<table>
<thead>
<tr>
<th>Value of ( J ) for different exponents</th>
<th>MEPSO</th>
<th>DE</th>
<th>G3 with PCX</th>
<th>GA in Mastorakis et al. (2003)</th>
<th>NN-based method Mladenov and Mastorakis (2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_1 )</td>
<td>60.3923 ± 0.0054</td>
<td>98.5513 ± 0.0327</td>
<td>95.7113 ± 0.382</td>
<td>96.7635 ± 0.8742</td>
<td>140.6201</td>
</tr>
<tr>
<td>( J_2 )</td>
<td>9.0005 ± 0.0323</td>
<td>11.9078 ± 0.583</td>
<td>10.4252 ± 0.989</td>
<td>10.0342 ± 0.0663</td>
<td>14.4182</td>
</tr>
<tr>
<td>( J_4 )</td>
<td>0.5039 ± 0.00054</td>
<td>2.9613 ± 0.0344</td>
<td>0.5732 ± 0.1324</td>
<td>0.6346 ± 0.0154</td>
<td>1.2163</td>
</tr>
<tr>
<td>( J_8 )</td>
<td>0.0058 ± 0.0001</td>
<td>0.2903 ± 0.0755</td>
<td>0.0178 ± 0.0396</td>
<td>0.0091 ± 0.0014</td>
<td>0.0698</td>
</tr>
</tbody>
</table>

| Table 4 |
Results of unpaired \( T \)-tests on the data of Table 3

<table>
<thead>
<tr>
<th>Cost function</th>
<th>Std. error</th>
<th>( t )</th>
<th>95% conf. intvl.</th>
<th>Two-tailed ( P )</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_1 )</td>
<td>0.195</td>
<td>179.313</td>
<td>-35.447 to -34.656</td>
<td>&lt; 0.0001</td>
<td>Extremely significant</td>
</tr>
<tr>
<td>( J_2 )</td>
<td>0.221</td>
<td>6.3440</td>
<td>-1.8516 to -0.9557</td>
<td>&lt; 0.0001</td>
<td>Extremely significant</td>
</tr>
<tr>
<td>( J_4 )</td>
<td>0.030</td>
<td>3.1041</td>
<td>-0.1518 to -0.0319</td>
<td>0.0036</td>
<td>Very significant</td>
</tr>
<tr>
<td>( J_8 )</td>
<td>0.0001</td>
<td>8.0000</td>
<td>-0.0035 to -0.0021</td>
<td>&lt; 0.0001</td>
<td>Extremely significant</td>
</tr>
</tbody>
</table>
binary encoded GA suggested in Mastorakis et al. (2003) on the design of a circular symmetric zero-phase low pass filter according to the user specification summarized in Eq. (14). All the algorithms discussed here have been developed from scratch in Visual C++ platform on a Pentium IV, 2.2 GHz PC, with 512 KB cache and 2 GB of main memory in Windows Server 2003 environment. The graphs and figures have been obtained using MATLAB 6.5.

Two aspects of the algorithms have been investigated in this work, firstly, their accuracy and then their speed of convergence. For comparing the speed of different algorithms the first thing we require is a fair time measurement. The number of iterations or generations cannot be accepted as a time measure as the algorithms perform different amount of works in their inner loops and also they have different population sizes. We have used the number of fitness function evaluations as a measure of time. The advantage of measuring complexity by counting the function evaluations is that there is a strong relationship between this measure and the processor time as the function complexity increases.

To judge the accuracy of the algorithms, we firstly run all of them (except the neural network based one reported in Mladenov and Mastorakis, 2001) for 50,000 FEs (Function Evaluations). Each algorithm is run independently (with a different seed for the random number generator in every run) for 20 times and the mean best \( J \) value as well as mean values of the parameters obtained along with the standard deviations have been reported in Tables 2 and 3. The notation \( J_0 \) has been used to denote two sets of experiments performed with the value of \( J \) obtained using exponent \( b = 2 \) and 4. Table 4 summarizes the results of the unpaired \( t \)-test on the \( J \) value (standard error of difference of the two means, 95% confidence interval of this difference, the \( t \) value, and the two-tailed \( P \) value) between the best and next-to-best results in Table 2. For all cases in Table 3, sample size = 20 and degrees of freedom = 38. Next we select four threshold values of \( J \) for \( b = 1, 2, 4 \) and 8 (based on the data of Table 3 and report the average number of FEs required by the optimization algorithms (over 20 independent runs) in order to reach the threshold in Tables 5 and 6. This gives a measure of the relative speed of the algorithms over this problem.

Table 5

<table>
<thead>
<tr>
<th>( J_b ) cut-off value</th>
<th>Mean no. of FEs required</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_1 = 70.00 )</td>
<td>26372.50</td>
</tr>
<tr>
<td>( J_2 = 11.00 )</td>
<td>32023.60</td>
</tr>
<tr>
<td>( J_4 = 0.80 )</td>
<td>24903.75</td>
</tr>
<tr>
<td>( J_8 = 0.10 )</td>
<td>30028.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DE</th>
<th>G3 with PCX</th>
<th>GA in Mastorakis et al. (2003)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_1 = 70.00 )</td>
<td>27736.25</td>
<td>33627.80</td>
<td>34637.05</td>
</tr>
<tr>
<td>( J_2 = 11.00 )</td>
<td>35237.95</td>
<td>39283.95</td>
<td>39291.45</td>
</tr>
<tr>
<td>( J_4 = 0.80 )</td>
<td>30483.80</td>
<td>40394.65</td>
<td>34243.85</td>
</tr>
<tr>
<td>( J_8 = 0.10 )</td>
<td>32948.55</td>
<td>40056.48</td>
<td>38291.45</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>( J_b ) cost function</th>
<th>Value of the figure of merits used</th>
<th>MEPSO</th>
<th>DE</th>
<th>G3 with PCX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum error of frequency response in pass band</td>
<td>0.005123 (0.0000021)</td>
<td>0.029105 (0.0000023)</td>
<td>0.009462 (0.0000028)</td>
<td></td>
</tr>
<tr>
<td>Maximum passband magnitude ripple</td>
<td>0.094625 (0.0000028)</td>
<td>0.329395 (0.0000046)</td>
<td>0.192182 (0.0000046)</td>
<td></td>
</tr>
<tr>
<td>Minimum stopband attenuation (dB)</td>
<td>29.19826 (2.66292)</td>
<td>34.18292 (2.66292)</td>
<td>34.18292 (2.66292)</td>
<td></td>
</tr>
<tr>
<td>Maximum magnitude of the poles</td>
<td>0.846325 (0.0002918)</td>
<td>0.891718 (0.0006752)</td>
<td>0.918192 (0.0006752)</td>
<td></td>
</tr>
</tbody>
</table>

...
Fig. 3. Desired amplitude response $|M_d(\omega_1, \omega_2)|$ of the 2D filter.

Fig. 4. Amplitude response $|M(\omega_1, \omega_2)|$ of the 2D filter using MEPSO with $b = 2$.

Fig. 5. Amplitude response $|M(\omega_1, \omega_2)|$ of the 2D filter using DE with $b = 2$.

Fig. 6. Amplitude response $|M(\omega_1, \omega_2)|$ of the 2D filter using G3 with PCX with $b = 2$.

Fig. 7. Amplitude response $|M(\omega_1, \omega_2)|$ of the 2D filter using in the method of Mastorakis et al. (2003) with $b = 2$.

Fig. 8. Amplitude response $|M(\omega_1, \omega_2)|$ of the 2D filter using in the method of Mladenov and Mastorakis (2001) with $b = 2$. 
Figs. 3–8 show the ideal low-pass filter frequency response and the frequency response of the filter designed using the competitive algorithms used here. All these figures are drawn from the results obtained using minimization of $J_2$ for 50,000 FE. Fig. 2 shows the progress of the optimization algorithms towards the optima of the search space with no. of function evaluations.

Finally in Fig. 9, we try to catch a glimpse of the performance of the designed low pass filters when applied to remove Gaussian noise from the renowned $256 \times 256$ gray scale image of Lenna. We here study one generic application of the low pass 2D filters in the de-noising of digital images (Gonzales and Woods, 1992). Fast Fourier transform (FFT) is performed on the original image using the MATLAB image processing toolbox. The filter transfer function is multiplied with the image in frequency domain and then the product is subjected to inverse FFT (IFFT) for obtaining the filtered image in a presentable form.

6. Discussion

A closer look at Figs. 3–8 reveals that the MEPSO algorithm used by us yields a better approximation of the desired response as compared to works presented in Mladenov and Mastorakis (2001) or Mastorakis et al. (2003) and yields a considerably good filter response with a smaller number of FEs as evident from Fig. 1. The ripple in the stop-band of Fig. 4 is much lesser as compared to Figs. 5–8. This leads to the filtering of a 2D filter with minimum distortion. The fact can be felt by taking a look

![Fig. 9. Result of filtering with the designed filter: (a) Original image “lenna”. (b) Image corrupted with Gaussian noise of mean $= 0$, variance $= 0.005$. (c) Filtering with the MEPSO-based method. (d) Filtering with DE based method (e) filtering with G3 with PCX-based method (f) filtering with GA method as in Mastorakis et al. (2003).](image-url)
at the de-noised image of Fig. 9(c) obtained with the MEPSO-based low-pass filter. This image preserves most of the details of the original one although considerably suppresses the Gaussian noise. As evident from the t-test results on the data of Table 3, the PSO-based method beats its nearest competitor in a statistically significant manner. The comparison also reflects the fact that the G3 with PCX model in Deb et al. (2002a), used by us, performs much better than the GA used in Mastorakis et al. (2003) in terms of accuracy. The possible reason may be that Binary coded GA has problem due to discretization of the search space. The speed of reaching the optimum is comparable for PSO and DE but poorer than the GAs as both PSO and DE uses very primitive mathematical operations. Especially, the main body of PSO algorithm is surprisingly simple (takes 4–5 lines to code in a standard language like C). The performance of DE is to some extent bleak in terms of accuracy. We believe DE can be improved for difficult optimization problems, like the present one, by suitably tuning its scale factor and the crossover rate.

7. Conclusion

In this paper, a recently developed optimization algorithm, borrowed from the realm of swarm intelligence, has been applied to the real-world problem of designing 2D zero-phase recursive filters. The filter thus obtained has a reasonably good stability margin (we have incorporated the stability criteria as constraints to the minimization task). Our method leads to a simpler filter since in practice we have realized a factorable denominator. Compared to the methods described in Mladenov and Mastorakis (2001) and Mastorakis et al. (2003), which, to our knowledge, are the most recent and the best-known methods to date, the algorithm used here yields a better design in considerably less time. The MEPSO algorithm has also shown to meet or beat two other very well-known evolutionary computing techniques (DE and a real-coded GA) in a statistically significant manner.

References